GCE
AFIT/~~GE~~/ENG/91D-7

SPEECH RECOGNITION USING MULTIPLE FEATURES AND MULTIPLE RECOGNIZERS

THESIS

Thomas F Rathbun

Captain, USAF

AFIT/GCE/ENG/91D-7

REPORT DOCUMENTATION PAGE	Form Approved OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	3 December 1991	Master's Thesis

4. TITLE AND SUBTITLE Speech Recognition Using Multiple Features and Multiple Recognizers	5. FUNDING NUMBERS
6. AUTHOR(S) Thomas F. Rathbun, Capt, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) School of Engineering Air Force Institute of Technology Wright-Patterson AFB, OH 45433	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr Timothy Anderson AL/CFBA Wright-Patterson AFB, OH 45433	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution unlimited	

13. ABSTRACT (Maximum 200 words)

The purpose of this ~~study~~ [illegible] is to demonstrate the feasibility of using multiple features and multiple recognizers to perform isolated word recognition. This is accomplished by performing multiple independent recognition tests and fusing the results together to get a single recognition result.

The speech data is recorded and each word is extracted into a separate file. Eight features are calculated for each word. The features are calculated on 512 sample time slices and produce 16 component vector outputs. The three recognizers use the eight features to produce a total of 24 error distance lists. These lists are then fused together by adding the error values corresponding to each word. The word with the smallest fused error value is declared the recognition winner.

Talker dependent and independent tests were run on a word set of zero through nine and A through Z. The talker dependent tests achieved accuracies between 87% and 100% depending on the talker. The talker independent tests achieved accuracies between 81% and 97%.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Speech, Speech Recognition, Dynamic Time Warping, Multi-featured fusion	311
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GCE
AFIT/~~GE~~/ENG/91D-7

SPEECH RECOGNITION USING MULTIPLE FEATURES AND MULTIPLE RECOGNIZERS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Thomas F. Rathbun, B.S.C.S.E.

Captain, USAF

December, 1991

Preface

The purpose of this effort is to explore the use of multiple features and multiple recognizers for recognizing isolated speech. Multiple recognitions are performed independently for each unknown word. The results are then fused together using feature fusion to determine a single recognition result.

In performing this study, I am deeply indedted to my wife whose encouragement helped me do my best, and whose timely distractions helped me to keep everything in perspective.

Thomas F. Rathbun

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this study is to demonstrate the feasibility of using multiple features and multiple recognizers to perform isolated word recognition. This is accomplished by performing multiple independent recognition tests and fusing the results together to get a single recognition result.

The speech data is recorded and each word is extracted into a separate file. Eight features are calculated for each word. The features are calculated on 512 sample time slices and produce 16 component vector outputs. The three recognizers use the eight features to produce a total of 24 error distance lists. These lists are then fused together by adding the error values corresponding to each word. The word with the smallest fused error value is declared the recognition winner.

Talker dependent and independent tests were run on a word set of zero through nine and A through Z. The talker dependent tests achieved accuracies between 87% and 100% depending on the talker. The talker independent tests achieved accuracies between 81% and 97%.

Recommendations for future efforts include testing different features, adding more recognizers, and using a more sophisticated fusion scheme. Neural networks should prove helpful for the latter.

SPEECH RECOGNITION USING MULTIPLE FEATURES AND MULTIPLE RECOGNIZERS

I. Introduction

Automatic Speech Recognition (ASR) is the ability of a computer to recognize a word from an acoustic signal. The human ear and brain perform this type of recognition with incredible speed and precision. Even though computers do not possess this same ability, some progress is being made. Researchers and engineers are continuing their 40 plus years of effort to duplicate the human speech recognition process. To date, there are commercial ASR systems available, but from a human factors stand point, many are limited or not practical.

The benefits of a human-like ASR system are almost limitless. Governments and companies everywhere will benefit as the productivity of the everyday computer user increases. The military could better improve the man-machine interfaces on many of its weapons systems. Handicapped persons could be better integrated into society.

ASR systems hold great promise for the future of society. Their development has been long in coming and there is still much work to be done. Although researchers may be progressing in small steps, a truly robust ASR system will one day be a reality.

1.1 Background

In the 1800s, Fourier proved that any repeating signal can be represented by an infinite sum of sine or cosine waves. The 1940s saw the Fourier transform applied to speech after the invention of the spectrogram machine (1). This invention gave speech researchers their first look at the frequency components of speech, enabling them to begin the quest for automated speech recognition. In the late 1940s, following the invention of the computer, scientists automated the calculation of Discrete Fourier Transform (DFT) coefficients (2). The Fast Fourier Transform (FFT) algorithm, developed in the 1960s, executes in O(nlog(n)) time and improves the speed of the DFT algorithm.

Early attempts to recognize speech were unsophisticated. Since the number of fundamental speech sounds in any language is finite, researchers attempted to map FFT coefficients to fundamental sounds of the language. This attempt failed when a consistent mapping between spectrogram vectors and fundamental speech sounds could not be found (2).

In the 1970s, researchers used pattern recognition techniques to match the input speech with stored patterns of speech. A recurring problem in speech pattern matching is the variation in the length of spoken words. Generally, the same person will not say the same word at the same rate (1). Thus, there is a need for the words in a speech recognition system to be time normalized. The most successful technique for time normalization is an algorithm called Dynamic Time Warping (DTW). Researchers such as Itakura-1975, White and Neely-1976, and Sakoe and Chiba-1978, developed DTW algorithms (3:297). DTW is sometimes called dynamic programming because it is implemented with dynamic programming techniques (3:299).

Many new techniques for speech recognition emerged in the 1980s. The Kohonen neural network, used to cluster similar speech sounds, and multilayer perceptron neural networks, used to segment and classify speech, were the subjects of extensive research at AFIT (4)(5)(6). The Hidden Markov Model (HMM) is used to model speech statistically and is the subject of much research in speech recognition.

1.2 Problem Statement

The purpose of this thesis is to develop and implement a multi-feature multi-recognizer system for word recognition of acoustic signals.

1.3 Research Objectives

To develop a modular feature independent recognition software system.

To develop multiple features to use for recognition.

To develop a feature image pattern recognition algorithm for speech.

1.4 Research Questions

What is the recognition accuracy for each feature and each algorithm?

What is the effect on the feature fusion recognition accuracy when adding additional features?

What is the effect on recognition accuracy when using different types of pattern recognition algorithms?

What is the recognition accuracy of the two dimensional pattern recognition algorithm and does it help in multi-feature fusion?

1.5 Definitions

Features of Speech Characteristics of speech are derived from the acoustic wave form. Typical features include the Fast Fourier Transform, formants, linear cepstrums, Mel cepstrums, linear predictive power spectrums, and using the difference between time slices of the aforementioned features (3:293).

Dynamic Time Warping A pattern matching process designed to eliminate the non-linear time difference when comparing spoken words. DTW produces a distance metric relating the similarity of a word to that of a referenced word (3:297).

Phoneme The smallest unit of sound used to describe or characterize spoken language. According to Rabiner, there are approximately 42 phonemes used to characterize American English.

Coarticulation "The overlapping of phonetic features from phone to phone (3:92)."

Frication The forcing of air through a restricted vocal tract to produce speech. This restriction causes turbulent airflow, therefore generating broad band noise (3:66). Voiced Frication occurs when the vocal cords are on and unvoiced Frication occurs when the vocal cords are off. An example of a voiced fricative is the 'z' in 'zero'. An example of an unvoiced fricative is the 'k' in 'cat' (6:5).

Plosive A temporary, complete closure of the vocal tract during speech. This closure causes increased pressure in the vocal tract, which, when released, produces a clean, sharp sound (3:87). The 'p' in 'upper' is a Plosive.

1.6 Assumptions

This thesis used a VAXStation II with a DSC analog to digital converter, a VAX 4000, and a VAX 6000 model 420. Each computer runs the VMS operating system with network access using DECNET.

1.7 Scope

This thesis will concentrate on developing many different features, many different pattern recognition algorithms, and investigate how they can all be used together for identifying acoustic signals

1.8 Summary

The development of a human-like ASR system is a worthwhile effort. As outlined above, the benefits of an ASR system are enormous. However, the task is neither simple nor success guaranteed. By studying the human speech recognition system, researchers may discover the means to create a human-like ASR system. Chapter 2 will give a cursory look of speech recogntion, including topics germain to this thesis.

II. Literature Review

2.1 Introduction

Research in the speech recognition field is proceeding at a rapid pace. The ever increasing availability of fast computers and general purpose software allows researchers to test more theories in less time. Large speech databases exist on CD ROM eliminating the need for researchers to collect their own speech data. This reduces the cost and speeds up the research. This chapter will focus on current research in the speech recognition field that relates to this thesis.

2.2 Human Speech Communication Process

Speech communication involves both speech generation and speech recognition. To develop a more usable ASR system, both processes need to be better understood. Researchers study the human speech generation process in hopes of determining how the brain encodes information to the acoustic signal. Likewise, researchers study the speech recognition process by examining the mechanisms of the ear. Their goal is to understand how the brain decodes lingual information and to duplicate the process.

2.2.1 Speech Generation

The organs involved in speech generation are the lungs, the larynx, and the vocal tract. The larynx (see figure 1) is the set of muscles and cartilage that houses and controls

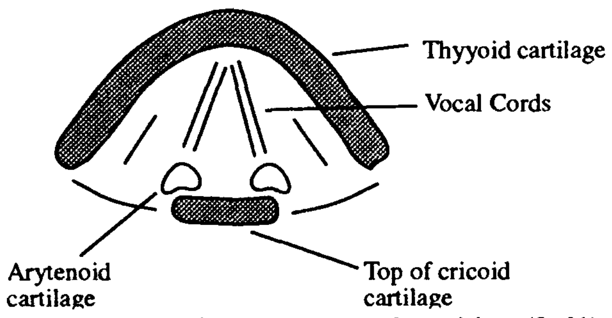

Figure 1 The larynx vocal cords open to breathing (3:61)

the vocal cords. The space between the vocal cords is the glottis. The frequency of the vocal cord vibration is called the glottal frequency. The vocal tract, composed of the nasal and oral cavity, the nasal and oral pharynx, and the laryngeal pharynx, is the acoustical tube that modulates the glottal output. Figure 2 shows the vocal tract and surrounding organs.

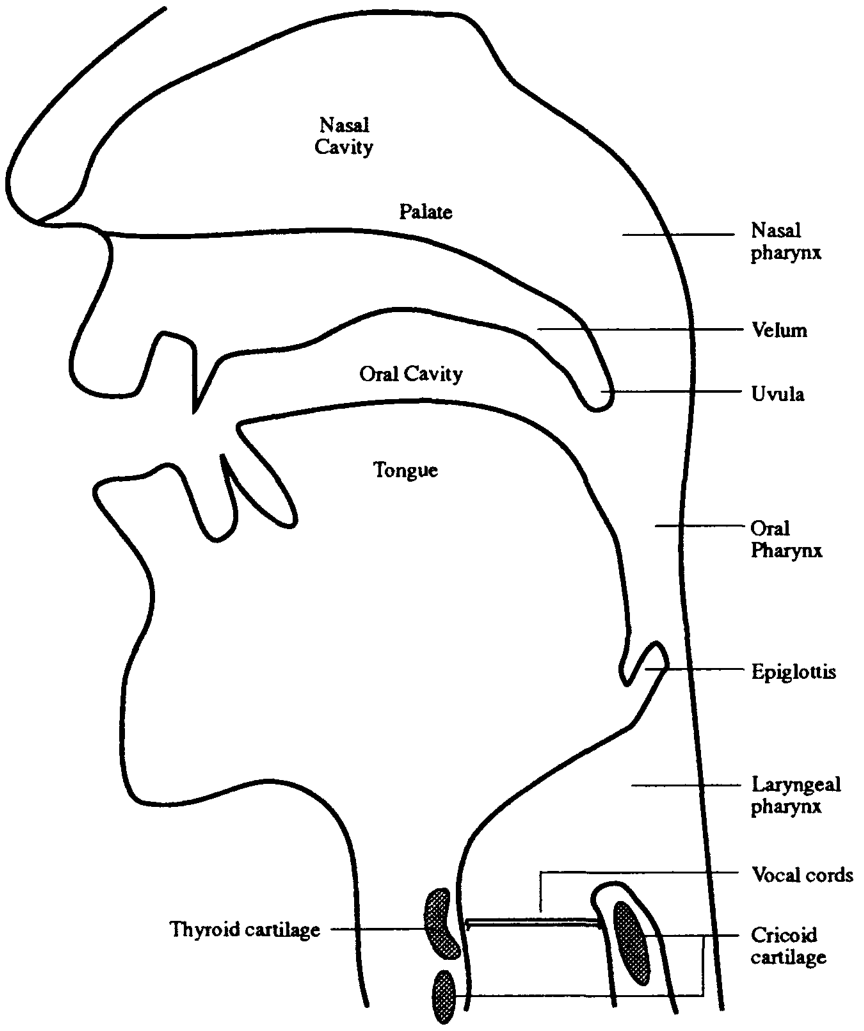

Figure 2 The tract and the surrond organs (3:63)

Speech generation is divided into two parts, excitation and modulation. Excitation generally takes place in the glottis, but sometimes occurs in other areas of the larynx. “Modulation is done by the various organs of the vocal tract. (3:64)”

Like the lips of a trumpeter, the vocal cords vibrate when air is forced through them. This is the most important form of excitation and is called phonation. However, when one whispers, air is not forced through the vocal cords. This means excitation must take place elsewhere in the larynx. Figure 3 shows the configuration of the larynx during phonation

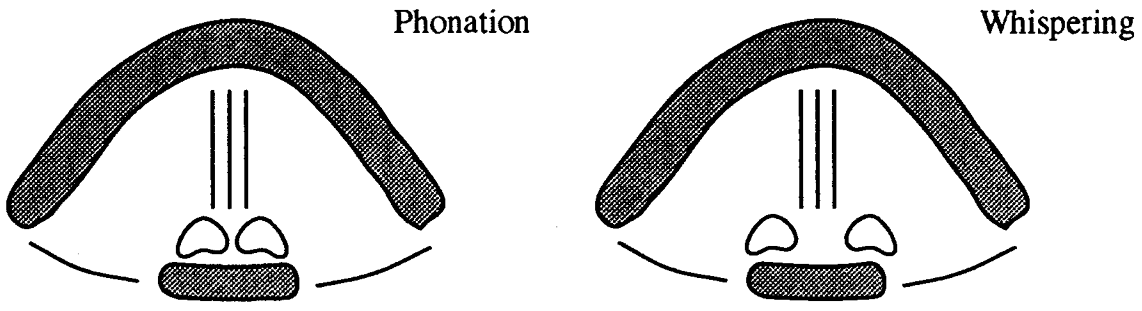

Figure 3 The vocal cords in various positions (3:65)

and whispering.

Modulation is akin to what linguists call articulation. When the shape of the vocal tract changes, the glottal output is encoded with linguistical information. The vocal tract, like any acoustical tube, has natural resonances.

> These natural resonances are called formants, and they are the single most important way of modulating the voice - they account for all vowels and some of the consonants, and we will see that they also provide crucially important information about the rest of the consonants as well. (3:66)

Figure 4 shows the shape of the vocal tract when modulating the vowel sounds AH, EE and OO. Additional types of modulation are interruptions and the addition of wide band noise. (3:59-66)

2.2.2 Speech Recognition

The ear consists of three main regions - the outer ear, the middle ear, and the inner ear. See Figure 5. The pinna and external auditory meatus make up the outer ear. The pinna is the protector of the meatus and may provide a person with directional cues. The meatus is a uniform tube that directs sound waves to the ear drum.

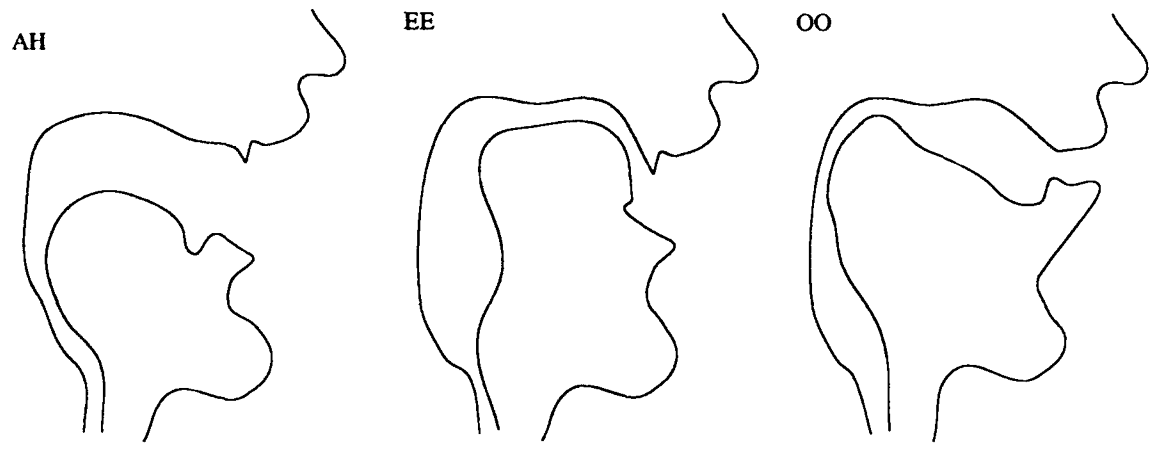

Figure 4 The vocal tract modulating the sounds AH EE OO

The middle ear is the air filled region that contains the eustachian tube and the ossicles. The eustachian tube allows the middle ear to be at a pressure equal to that of the outside world. The ossicles, composed of three tiny bones, connects the eardrum to the cochlea and provides impedance transformation and amplitude limiting. Impedance transformation increases the amount of acoustical energy that is transferred from the eardrum into the cochlea by 15:1. Amplitude limiting protects the cochlea from high decibel sounds by limiting the transfer of high amplitude sound waves.

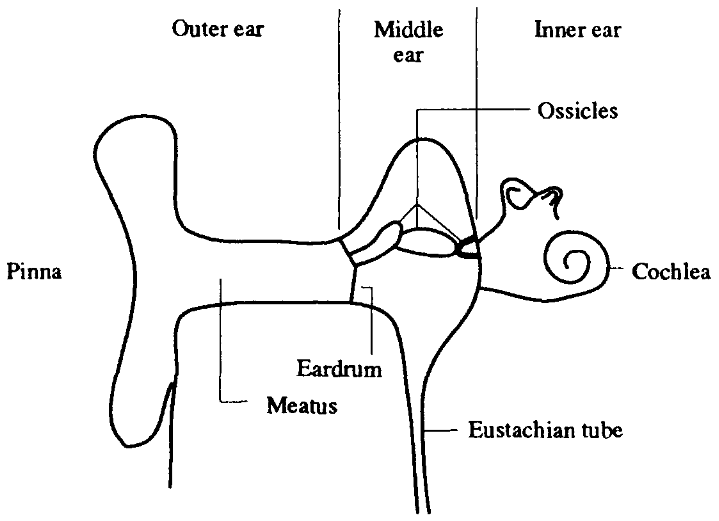

Figure 5 The three regions of the ear (3:67)

The inner ear has three parts, but only the liquid filled cochlea contributes to the speech recognition process. A cross section of the cochlea is shown in Figure 6 . The coch-

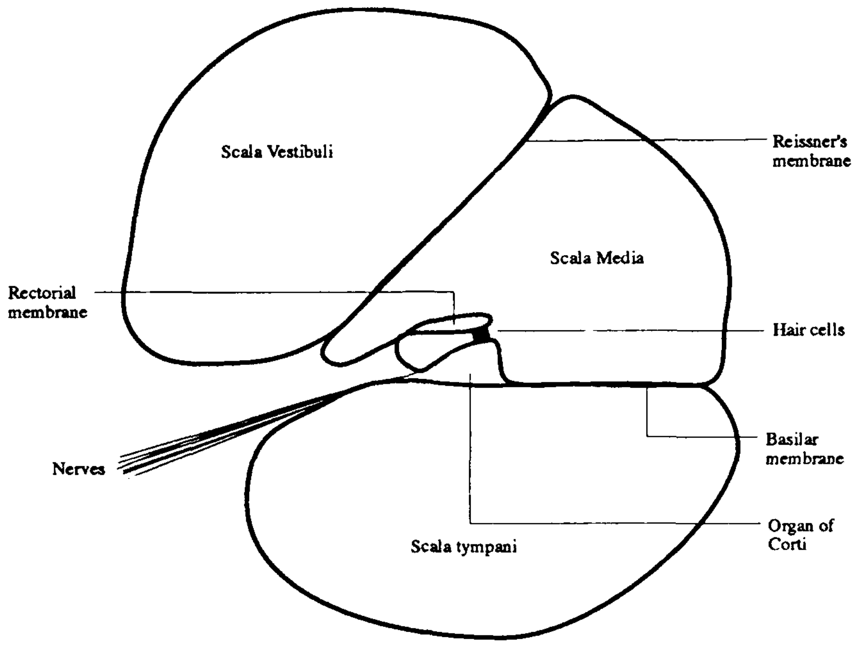

Figure 6 Cross section view of the cochlea (3:69)

lea, which is divided by the basilar membrane, resembles a snail. The acoustical signal enters the cochlea, propagates through the liquid of the Scala Tympani, passes to the opposite side via the Helicotrema, then propagates back through the Scala Vestibuli before exiting. The helicotrema is the end of the cochlea where the basilar membrane stops. The propagation of the sound waves through the cochlea causes the basilar membrane to vibrate. The hair cells connected to Organ of Corti move with the vibrations and cause an electrical potential in the nerve cells. The frequency information is transmitted to the brain by the firing of these nerve cells.

The cochlea separates sound frequencies spatially along the length of the basilar membrane. Each sound wave causes a specific area of the basilar membrane to vibrate at its resonate frequency. Since the sound frequencies are separated, individual frequency information can be passed to the brain stem where further processing occurs. The frequency information is then sent to the auditory cortex on the opposite side of the brain. (3:67-71)

2.3 Classifications of Speech Recognition

Speech recognition systems are classified by the type of speech input and speaker dependency (1). The size of the vocabulary and the recognition accuracy are the measures of performance for a speech recognition system (1).

2.3.1 Speech Input

Isolated word and continuous word are the type of inputs that are used in speech recognition systems. Isolated-word recognition systems generally have a higher recognition accuracy than continuous word systems (7:27).

2.3.1.1 Isolated Word Recognition

Isolated word recognition requires inputting one distinct word at a time. The input can be either a single word or a stream of words with a pronounced pause between each word. This pause helps eliminate speaker coarticulation and makes word boundary determination relatively easy (3:293). After determining the boundaries, a pattern matching algorithm is used for word recognition.

Recognit.on in isolated-word recognition systems is usually done at the word level (3:295). The system stores a library of spoken words and uses a pattern matching algorithm to determine similarity between input words and the library words. The system selects the library word that bears the highest degree of similarity to the input word. A commonly used pat.ern rec.ognition technique is Dynamic Time Warping (DTW). The DTW algorithm is computationally intensive, which, depending on the power of the computer, can limit the size of the vocabulary the recognizer can handle. Another algorithm that offers similar recognition accuracy at a fraction of the computational cost is the Hidden Markov Model (HMM) (3:307).

2.3.1.2 Continuous Word Recognition

Generally, when people speak, there is not an easily identifiable boundary between words and many words are not fully articulated (3:318-319). Therefore, the two major problems in continuous speech recognition systems are endpoint detection and coarticulation (3:318-319).

A modified form of the DTW algorithm can be used in continuous speech recognition systems (6:13). The modified DTW algorithm compares library words to the sections of the input utterance. When a library word satisfactorily matches a section of the utterance, the word is recognized and some boundaries are identified. The algorithm continues until the entire utterance is recognized, but the task becomes easier after each boundary identification.

Another approach to continuous speech recognition is segmenting input speech into phonemes of a language (6:13). The phonemes are then combined to make words. Artificial neural networks are frequently used to segment speech into phonemes. Rule base systems and HMMs are used to recognize words from phonemes. This approach allows the system to have a larger vocabulary because template words need not be stored and neural networks are computationally fast (2).

2.3.2 Talker Input

Every voice is somewhat unique. Differences in the glottal frequencies and the size and shape of the vocal tract create the uniqueness in each voice. A child, for example, has a smaller vocal tract which causes a higher pitched voice. Thus, formant frequencies for the same phoneme differ between speakers and impact speech recognition (1).

2.3.2.1 Talker Dependent

Speaker dependent systems, whether isolated word or continuous speech, perform recognition on only one speaker's voice. Therefore, all words in the system's library must be input by the same person. These systems usually have higher recognition rates than independent speaker recognition systems do since they do not have to compensate for inter-speaker variances (7:27).

2.3.2.2 Taker Independent

Speaker independent systems, whether isolated word or continuous speech, perform recognition for multiple speakers. The word library for a speech recognition system can be made speaker independent by using one or more of the following techniques: store multiple copies of the same word from different speakers, create word templates to represent many speakers, or filter out the speaker dependencies in speech. To create a template for a word, corresponding time slices for the same word, spoken by different speakers, are averaged together. To make speech data somewhat speaker independent, the formants frequencies of each speaker are either normalized or used as ratios. Normalization can be done with respect to vocal tract range, Gerstman 1968; vocal tract length, Wakita 1977; or vocal tract distribution, Neuberg 1980 (3:305-306). Typically, only the first three formant frequencies are used in speech recognition system. Using the formant ratios formant 2/formant 1 and formant 3/formant 1, improved speaker independent speech recognition in tests done at AFIT (6:97).

2.3.2.3 Talker Adaptive

A speaker adaptive system is a speaker independent system that improves its accuracy with continued use. When a new speaker inputs words to the system, the system updates its library to improve the recognition accuracy. If a recognition mistake is made, the mistaken utterance is averaged in with the correct library word. This average can be either a true average or a weighted average. The weighted average will make the library word more closely represent the word as spoken by the new speaker. The true average represents all speakers equally. Public systems use the latter technique.

2.4 Speaker Recognition

Speaker recognition serves two purposes, speaker verification and speaker identification (3:332). An example of speaker verification is using a system to grant or deny access to a restricted area based on a voice sample. Speaker identification is a useful tool for policewhen trying to identify an individual from a phone call or a taped conversation. Using a person's voice for identification is not the same as using fingerprints, since a person's voice may change with respect to his age and health (7: 32).

Problems exist with this technology. A hostile speaker may try to disguise his voice (3:333). Poor signal to noise ratio in a speaker verification area may cause a high number of false rejections (3:333).

2.4.1 Text Free

A text free speaker recognition system uses any speech input from a person to recognize the speaker. This type of recognition usually requires between 30 and 60 seconds of input speech to make a recognition prediction (3:336-340) (7:32).

One approach to this problem is to use vocal tract modeling. Given enough speech data, the vocal tract can be modeled from the formant data. Then, new speech is compared against the library of vocal tract models. People with the ability to mimic others are successful in spoofing these systems (3:333).

Other systems try to detect nuances and mannerisms specific to a person's voice to determine recognition (3:333). Typical features of this recognition approach include (3:335):

1) pitch at selected points in words,

2) spectral characteristics of nasal constants,

3) spectral characteristics of selected vowels,

4) estimated slope of the excitation spectrum,

5) spectral characteristics of fricatives,

6) duration of selected vowels,

7) presence of prevoicing in a selected context.

In controlled speaker recognition tests, using the above 7 features plus 10 others, the error rate was 0 percent (3:335). The typical error rate for this type of system is less than 3 percent (3:335).

2.4.2 Text Specific

Unlike the text free systems, text specific speaker recognition systems require the user to input the same word or phrase each time to recognize the speaker. This approach is similar to speaker dependent, isolated-word speech recognition systems, and works well when used with the same speaker templates, but not so well when used with cross speaker templates (6:84)(7:27). A speaker recognition system stores a library of the same word or phrase as spoken by different speakers. Then, it compares the input word to the library words and creates a list of DTW distances. Since the speaker may not be in the library, the smallest DTW distance will not always determine the speaker. However, if the smallest DTW distance is significantly less than the other DTW distances, then it is likely that this distance will identify the speaker.

2.5 Speech Recognition Techniques

2.5.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a pattern matching technique that uses a distance value to indicate similarity. The smaller the value, the closer the similarity. The purpose of DTW is to discount the variable time in which people say the same word (1). DTW allows localized shrinking and stretching of parts of words to better determine if the input word and the library word are the same. If one speaker says “car” and another speaker says "caaaar," the speech recognizer, without DTW, will interpret the speakers as saying two different words. DTW shrinks the multiple "a" sound articulated by the second speaker to match the single "a" sound spoken by the first speaker. This creates a closer match and produces a smaller DTW distance. Figure 7 shows the effect of DTW in the time domain.

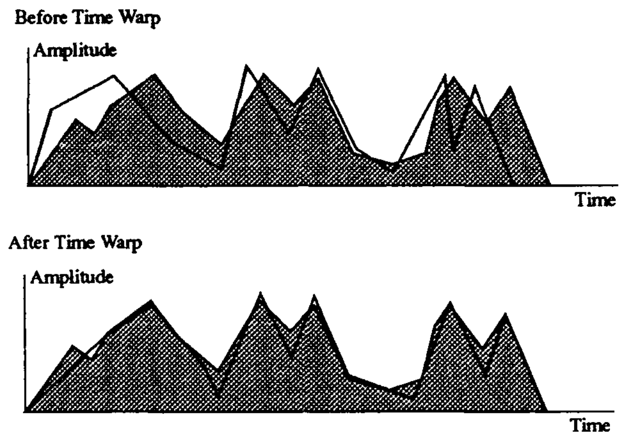

Figure 7 DTW, as represented in the time domain (7:29)

The shaded region represents the template which is stationary. The amplitude values of the line move in time to better match the template.

The DTW algorithm compares each point in the test word with every point in the library word. This produces a rectangle of point by point comparisons which resembles an error surface when viewed as a contour plot (1). Starting in the lower left hand corner, DTW looks for the shortest error path to the upper right hand corner. The DTW distance is the summation of all error values along the path (1). Figure 8 shows a hypothetical error surface that the DTW algorithm has to traverse. The size of the dots represent the size of error. A valley in the error surface that is along the main diagonal shows a strong correlation between the two words.

There are different approaches to finding the smallest path through the error surface. The classical approach is to allow movement in three directions — directly to the right, diagonally up to the right, or straight up (2). See Figure 9, left, for illustration.

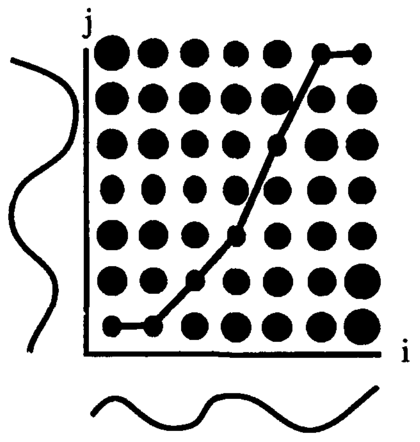

Figure 8 DTW error surface

Itakura's algorithm, removed the option of going straight up and added the option to go diagonally up two to the right (2). Itakura also limited going directly to the right to one time consecutively. This algorithm moves as a function of i, with i being the horizontal axis (3:301). See Figure 9, right.

Since DTW is computationally intense, the stopping criteria is important. In the worst case, DTW terminates when the error path reaches the end of the words. Other ad hoc stopping criteria are (3:300):

1) if the current accumulative distance is exceedingly large,

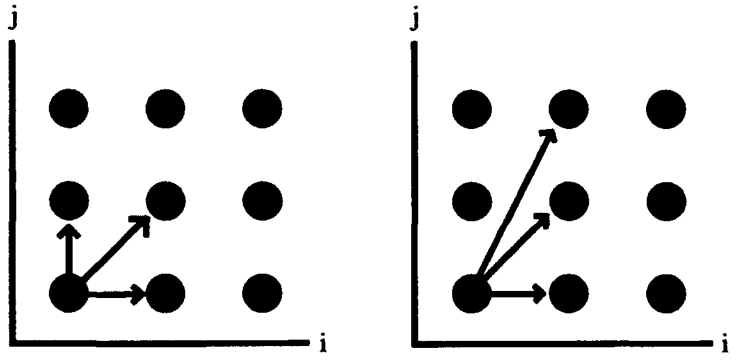

Figure 9 Two methods of traversing a DTW error surface

2) if the current accumulative distance is larger than the smallest DTW distance already obtained,

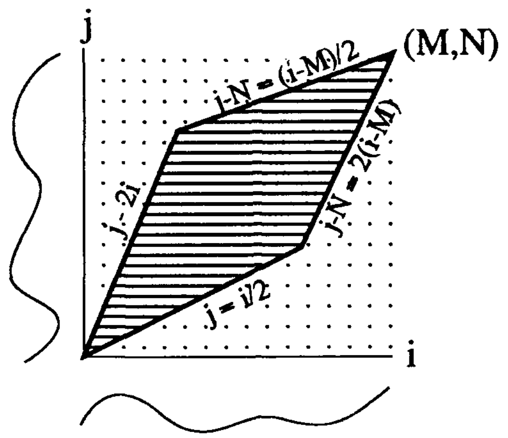

Figure 10 DTW parallelogram contraint (3:300)

3) if the error path wandered outside the parallelogram constraints, as shown in Figure 10 (3:300).

2.5.2 Hidden Markov Models

"A Hidden Markov Model is a doubly stochastic process for producing a sequence of observed symbols" (7:30). In the Hidden Markov Model (HMM), each word is modeled

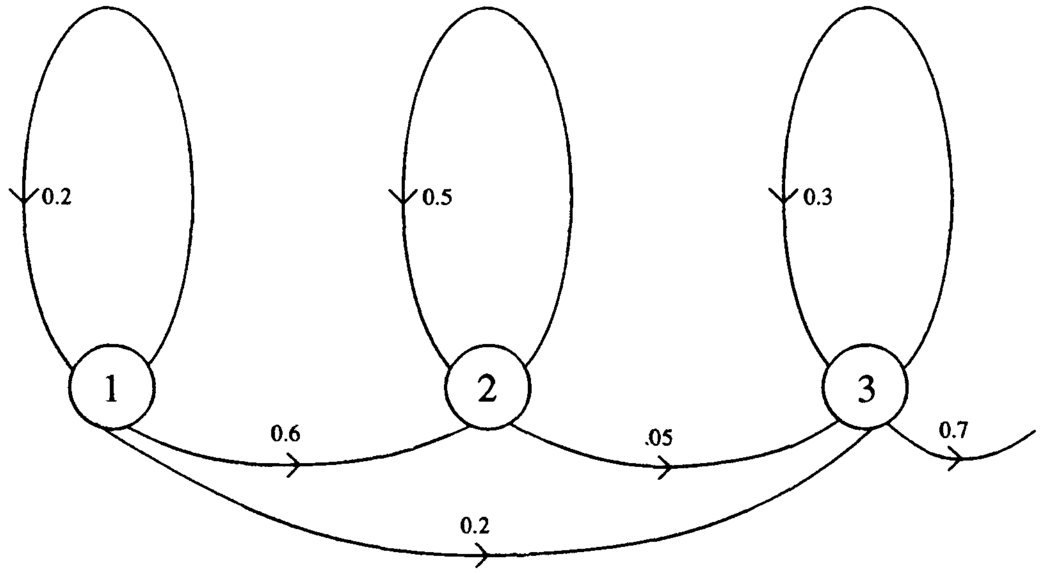

Figure 11 Finite State Model used in a HMM (7:30)

as a Finite State Machine (FSM). See Figure 11 for diagram. Each state is a phoneme or subphoneme sound unit with an identifying symbol. The links indicate the possible transitions and the probability of that transition occurring. A symbol that links back to itself represents a repeating sound. A link that skips the next symbol represents a word that is not fully articulated or has different pronunciations. With careful training, the FSM can add additional sound states that account for coarticulation. The probabilities are set using empirical data. The data changes with regional dialects.

In a recognition system, every word in the library is represented as an FSM model. The recognition process starts when the utterance is received. It is then segmented into a stream of symbols. Each FSM determines the probability of producing the identical symbol stream. The FSM with the highest probability of producing the input symbols is selected as the input word (3:310).

2.5.3 Artificial Neural Networks

Artificial Neural Networks (ANN) are a great boon for pattern recognition because they generate discriminate functions from the data (8). ANNs undergo a learning process where data is fed into the neural network and the interconnection weights are adjusted until the network can distinguish the different classes of data (8). If the data is separable, ANN's can produce discriminant functions that determine the classification of the data (8). Figure 12 shows a multi-layered perceptron neural network.

Speech recognizers use ANNs to segment speech data into phoneme or subphoneme sound units. The networks have an output node for each class of sound and enough input nodes to encompass the time interval of the digitized speech data.

2.6 Speech Recognition at AFIT

2.6.1 Barmore

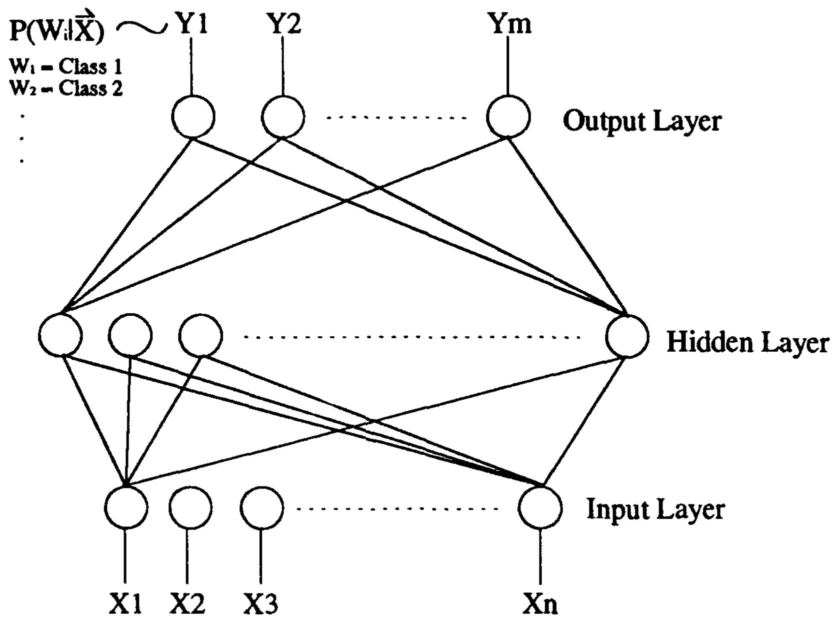

Figure 12 Multilayer Preceptron Neural Network

In 1988, Capt Barmore developed two types of speaker dependent speech recognition systems. The first type used two Kohonen neural networks, one to cluster sounds to nodes and the other to cluster words to nodes. The speech data is fed into the first network and a list of nodes representing the input data is produced. This list of nodes is fed into the second network, and it produces a node that represents the input word. Capt Barmore's second system replaced the second Kohonen neural network with a DTW algorithm that determines the input word.

Capt Barmore used a 10 word vocabulary to test his system. The first system's recognition accuracy was 96 percent on isolated words and 81 percent on connected words (4:5-1). The second system achieved accuracies of 99.1 percent and 90.7 percent respectively (4:5-1).

2.6.2 Recla

Capt Recla used the same architecture as Capt Barmore's second system, but added a second feature set. The two different features used were Linear Predictive Coding (LPC) and formants. The system library stored each word as an LPC file and as a formant file. The recognition system performed recognition twice, once for LPC and once for formants. Capt Recla then used multi-featured fusion to combine the outputs of the two recognition tests into a single output.

Capt Recla extended the vocabulary to the 70 words from the AFTI F-16 program for his recognition tests. Capt Recla achieved a recognition accuracy of 97.2 percent for isolated word and 70.8 percent for connected words (5:2-4,139).

2.6.3 Stowe

Capt Stowe added a third feature, the cepstrum, to Capt Recla's system. This system performs three separate recognition runs and uses multi-feature fusion to produce a single output prediction.

Capt Stowe extended the vocabulary to 86 words and tested both isolated and connected speech. He achieved accuracies of 98.7 percent and 70.3 percent respectively (6:73,79). Capt Stowe tested his system for speaker independence by using the same library for many different speakers. These tests produced poor results with many of the test runs achieving accuracies of less than 50 percent (6:84-95). In another attempt at speaker independence, Capt Stowe combined multiple speaker's words into one word templates. These templates are more representative of the population. Using the new template library, Capt Stowe achieved an accuracy of 82 percent on isolated speech (6:102).

2.7 Summary

This chapter is meant as a broad overview of speech recognition topics. Hopefully the reader can better appreciate the difficulty of the problem. Researchers are constantly coming up with new approaches to speech recognition but, to date, only small gains have been realized. Chapter 3 describes AFIT's facilities for speech recognition research.

III. System Environment

3.1 Introduction

The purpose of this chapter is to describe the environment for system development and operations. This thesis will develop an all new speech recognition system. The development of this system will be influenced by the types of computers available, their networking capability, and access to speech recording equipment. All equipment and software used for this development effort will be listed in Appendix B.

3.2 Speech Recording

In this thesis, all speech data will be recorded digitally using a Digital Sound Corporation model 200 (DSC-200) digital to analog converter. The speech data is entered using a noise reduction microphone, filtered with an 8 kHz anti-aliasing filter, and then digitized at 16 kHz. There are a few user setable settings on the DSC-200. All speech is recorded with the microphone gain on and the record level set at the third vertical line from the left.

The DSC-200 connects to CALVIN, a VAXStation II/GPX. The two programs that run on the VAX and operate the DSC-200 are RECORD and LISTEN. The RECORD program is used to record speech data. The user must supply a file name, the length of recording time, the sampling rate in component and mantissa form, and the channel information for the program to run. The digitized speech data is stored in the file name provided and the file is located in the user's directory on CALVIN. CALVIN's disk drives are relatively small and can hold only a few minutes of speech. The LISTEN program is used to playback recorded speech files. The user need only supply the filename for LISTEN to operate. The rest of the information, like sampling rate and channel information, is obtained from the header of the file. However, the user can distort the playback by specifying an alternate sampling rate for playback.

3.3 Hardware Environment

The number of AFIT computers and the network configuration never stays the same for more than a few days; therefore, no accurate diagram is possible. This thesis uses just a small portion of the available resources. The main software will be developed and run on HERCULES, a VAX 6000 model 420. HERCULES is a multiprocessor system with two, eight MIP processors. Also, HERCULES has a large disk farm managed by a hierarchical storage controller. Two RA60 removable disk drives are dedicated to this thesis. Each RA60 holds 270 megabytes of data. HERCULES is a member of a VAX cluster, meaning other VAXes in the cluster can be used for the same task as HERCULES. HERCULES is the fastest and is the primarily resource, but the other VAXes are suitable for executing batch jobs.

The second computer used in this thesis is CALVIN. CALVIN is a one MIP system used for recording the speech data. CALVIN's multiple windowing workstation capabilities are useful for interfacing with HERCULES.

DECNET is used to connect HERCULES and CALVIN on an AFIT subnet. DECNET is a network operating system which not only allows remote connects and file transfers, but also intra-network process communications. Thus, DECNET enables all networked VAX computers to appear as one large computer. This thesis makes use of this capability, allowing HERCULES and CALVIN to appear as one computing system.

3.4 Software Environment

The HERCULES operating system is VAX/VMS version 5.4 with DECNET phase IV. All software is developed using the VAX Ada version 2.2 programming language. Since the programs require extensive amounts of memory, all programs compile with the space optimation qualifiers.

The CALVIN operating system is VAX/VMS 4.7 with DECNET phase IV. CALVIN runs the VAX windowing software system. There are no compilers on CALVIN, but any VAX compiled code can run on CALVIN.

3.4 Tools

All the code is developed using the Language Sensitive Editor (LSE). The LSE contains templates for use with most programming languages. The advantage of using LSE is that it assists the programmer in entering code. This guarantees correct syntax for programming structures. For example, if the programmer types IF then control E, the following code segment will appear:

```
if {logical clause} then
{statements}...
[elsif]...
[else]
end if;
```

Statements in {} brackets must to be filled in, and statements in [] brackets do not have to be filled in. LSE has templates for all structures for the programming language and indents these structures for readability. Programs can be compiled inside the editor. When errors occur, the LSE splits the screen and shows the errors on the top and the code on the bottom.

3.6 Summary

This chapter presents an overview of the hardware and software environment used for the development of the new recognition system. The next chapter discusses in detail the design of the new recognition system.

IV. System Design

4.1 Introduction

This thesis will focus on developing an integrated software tool for testing speech recognition concepts. This tool will use some concepts from previous theses, but will be completely redesigned in ADA (4;5;6). This new software is called the AFIT Speech Recognition Tool (ASRT), pronounced assert. ASRT includes the capability to extract words from a digitized speech file, to calculate multiple features for each word, to perform word recognition using three different recognizers, to use multi-featured fusion to produce a single recognition result, and to combine multiple words into a single word template. The goal for this thesis is to show that a multi-featured, multi-recognizer system is robust enough to obtain high recognition accuracies for independent talkers.

This chapter discusses the complete design of ASRT. The areas discussed are the software design and the major subsystems. As stated earlier, this software is a complete redesign and it is hoped that ASRT will be used and improved upon in future AFIT theses.

4.2 Software Design

The design of the software is particularly important for meeting the goals of this thesis and for use in future theses. Sound software engineering techniques, such as a modular design, are crucial to maintaining and upgrading this software system.

The software system breaks down into the following seven areas:

1) Extraction operations

2) Database management

3) Feature operations

4) Recognition operations

5) Fusion operations

6) Template operations

7) Batch Interpreter

Fig''re 13 shows the interface between the six major subsystems in the interactive mode. In this mode, the user directs ASRT to perform all tasks using the menu commands of each subsystem. Although tedious, the interactive mode is designed for debugging and trying out different ideas. To perform large scale recognition tests, the system has a built in scripting language, which automates this process. The script interpreter is discussed in section 4.9.

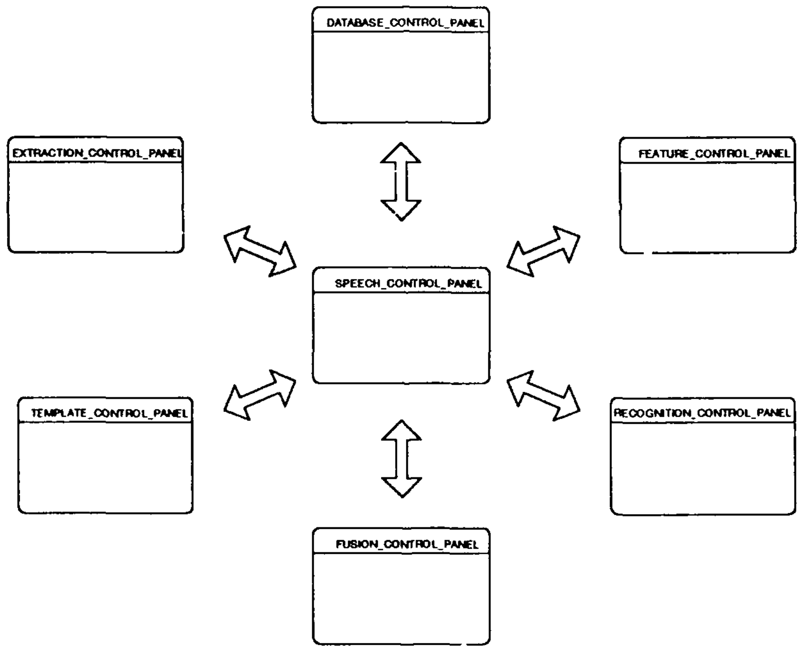

Figure 13 ASRT Software Subsystems

The software is designed in modular subsystems, using the powerful ADA package construct. To prevent the duplication of code, the software relies heavily on the use of generic packages and generic subprograms. Figure 14 shows the software layout in icons of packages and subprograms. Subprograms are in square boxes and packages are in rounded boxes. Subprograms resident inside packages which are visible to the user are shown in

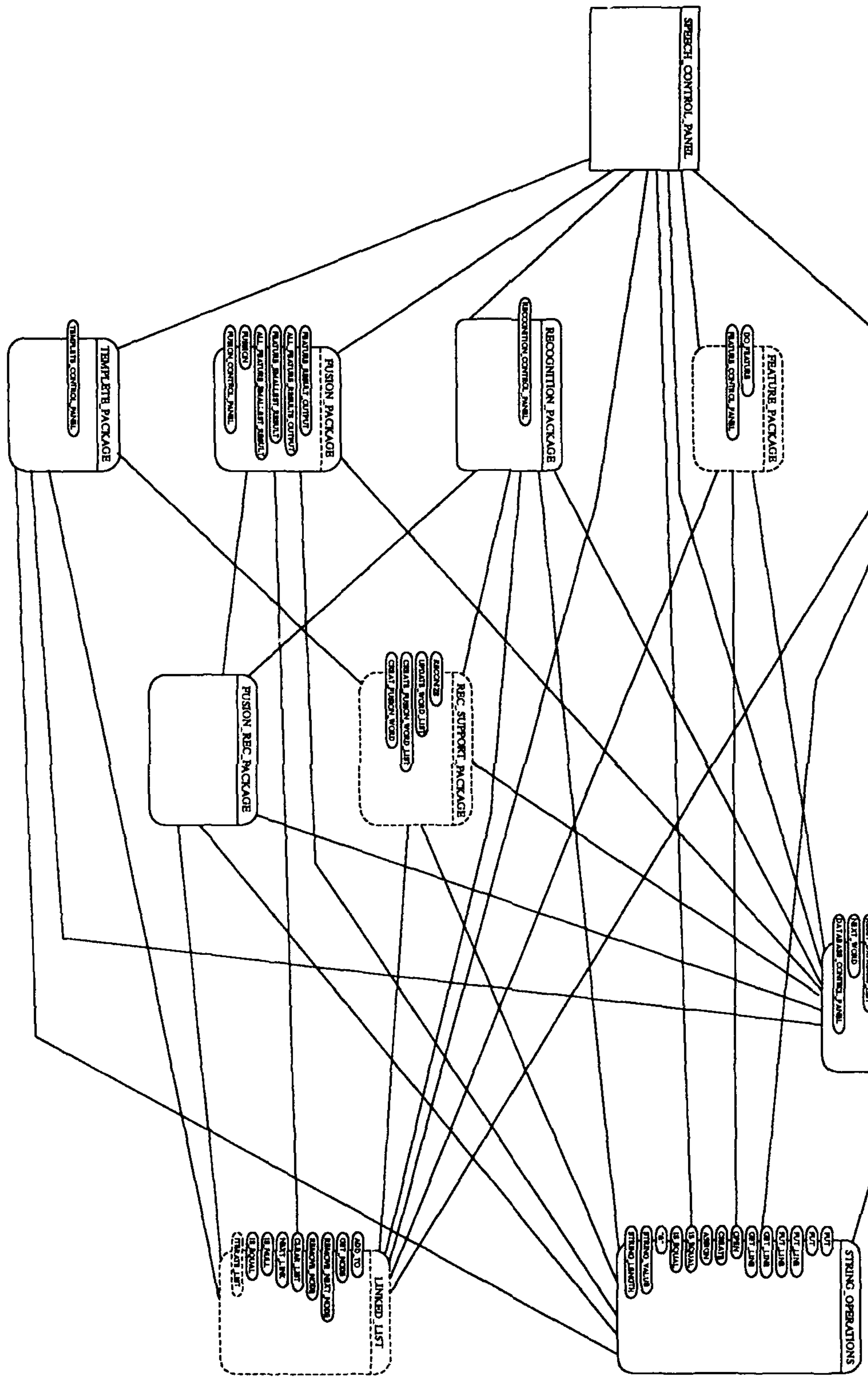

Figure 14 ASRT Softtware Package structure

oblong ovals. Generic packages and subprograms are indicated with dashed lines. Lines between packages indicate the package on the left is dependent upon the package on the right.

4.3 Extraction Operations

The EXTRACTION_PACKAGE contains all of the procedures and functions necessary for processing digitized speech files. Each digitized file contains one or more spoken words surrounded by noise. The goal is to separate the spoken words from the surrounding noise and store the words in a separate ASCII data file. This process consists of three steps: reading and translating the digitized speech file, calculating the Noise Distance Factor (NDF), and extracting the spoken words into separate files.

4.3.1 Reading Digitized Speech Files

The RECORD program, described in chapter 3, first stores a 512 byte header in the speech file, and then stores the speech data as 16 bit integers in 512 byte records. Ada duplicates this record structure by using the following type statement:

```
type SPEECH_RECORD_TYPE is array (INTEGER range 1..256) of SHORT_INTEGER.
```

An Ada short_integer is two bytes. The normal Ada GET procedure cannot read the speech file because it does not know the record structure. Another Ada input output package, SEQUENTIAL_IO, is a generic procedure that can be instantiated for any abstract data type. Therefore, if the file is opened as a sequential file, then the SEQUENTIAL_IO.READ procedure can be instantiated to read the file. The following instantiation incorporates the record structure into the SEQUENTIAL_IO package:

```
package  SPEECH_IO is new SEQUENTIAL_IO (SPEECH_RECORD_TYPE)
```

This instantiation allows the newly created SPEECH_IO.READ procedure to read one 512 byte record from the digitized speech file at a time. The first record is thrown away since it is header information. Then, each subsequent record has each data point converted from short_integers to floating point values which are stored in an ASCII data file.

The RECORD program runs on CALVIN and stores the speech files on CALVIN's disk drives. The ASRT software runs on HERCULES, but uses a DECNET link to read the digitized speech files from CALVIN. This remote file access helps speed up and automate the speech recognition processing of the ASRT system.

4.2.2 Classifying Speech Time Slices

The speech classification process is necessary for the extraction process to work. In essence, the extraction process is done as a two pass process. Pass one classifies each time slice as to how close it is to being noise. And pass two extracts the words and stores them in separate files.

The digitized speech file is sliced up into time slices consisting of 256 data points each, representing 16 ms of speech. The Linear Predictive Power Spectrum (LPPS) feature is calculated for each time slice. LPPS is used because it works well for speech recognition; this feature will be described in section 4.4.1. The Euclidean distance is calculated between the first time slice and all subsequent time slices. Then each Euclidean distance value is divided by the value of the average Euclidean distance between the first time slice and the next k - 1 time slice. This value is known as the Noise Distance Factor and is defined by equation 1.

$$NDF_i = \frac{\| LPPS_i , LPPS_1 \|_2}{\dfrac{\sum\limits_{j=2}^{K} \| LPPS_j , LPPS_1 \|_2}{K-1}} , \quad i = j+1, j+2, \ldots , N \qquad (1)$$

where

NDF is the Noise Distance Factor

LPPS is the Linear Predictive Power Spectrum

N is the number of time slices in a word

K is the number of time slices used to characterize the noise

A value of 9 was used for K, which assumes the first ten time slices consist of just noise data. The results of equation 1 represent how far each time slice is away from the noise. These values are truncated to integer values and stored in a linked list. Values over 9 are reduced to the value of 9. More detail on the speech characterization list and how it is used is provided in the next section.

4.2.3 Extracting Individual Words

The DSC records speech in predetermined time intervals. When speech is recorded, there is noise before and after each word. To speed up the recording of a large amount of speech data, multiple words are recorded into a single file. A typical speech file of three words is shown in Figure 15. The noise surrounding the words makes it necessary to extract each word into a separate file before further processing can occur. To extract the complete word in an automated fashion is not a simple process. A fricative at the start of the word may look like the surrounding noise. A plosive in the middle of the word can make one word look like two separate words. A word with a trailing plosive makes the end point difficult to determine, since the sound is offset from the rest of the word and has alow energy value. Figure 15 illustrates the above three conditions on a typical speech file of the words: six, upper, and front. An alternative to automated extraction is manually extracting each word. In manual extraction, the user determines the endpoints of the words. Manual extraction is slower but perhaps more accurate.

4.2.3.1 Automated Extraction

Pass two for the automated word extraction process uses two heuristic rules to determine the beginning of the word and the end of the word. These rules are applied to the

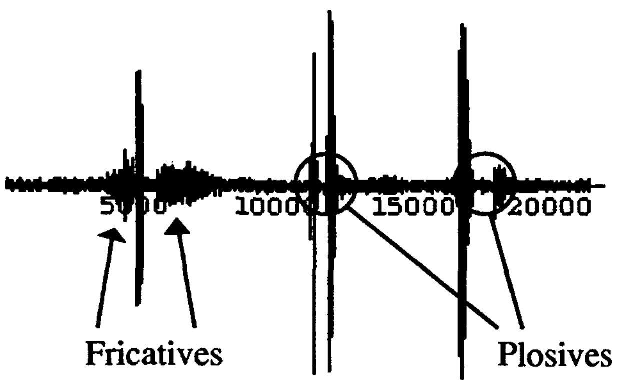

Figure 15 Digitized speech file of three words

speech classification list of pass one. The numbers used in the following heuristics can be adjusted by the user.

The beginning word heuristic is — if 4 time slices in a row have a NDF greater than 2, then the first data point of the first time slice is the start of the word. The ending word heuristic is — if 12 time slices in a row have a NDF less than 3, then the last data point of the last time slice classified as speech is the word's end point. All data points in between the two endpoints are written to a file to complete the extraction process. Figure 16 shows the extracted words from the file shown in Figure 15. Normally, the user specifies the file name for each word as it is extracted. However, two file name generating alternatives are available. The file names can be generated automatically with the following naming convention: UNKNOWN_WORD_1, UNKNOWN_WORD_2, ... UNKNOWN_WORD_#. Also, the file names can be generated from the list of the database words. This assumes that the first word extracted is first word in the list and so on. The method may not be very reliable.

4.2.3.2 Manual Extraction

Since automated procedures do not always work perfectly, a manual extraction program is also provided. The manual extraction program displays the speech classification list to the user. Figure 17 shows the classification list for the speech file in Figure 15. The numbers in the classification list are the NDF values from equation 1. This allows the user to

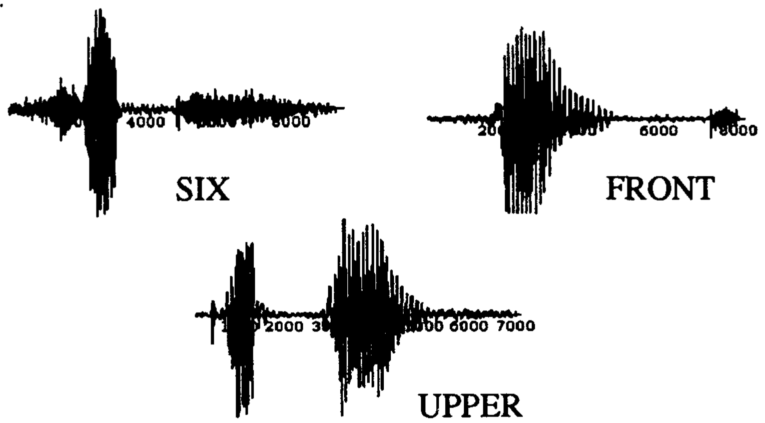

Figure16 Extracted digitized words

visually determine the rough location of the word's endpoints. The user tells the program how many time slices to delete, or how many time slices encompass the word. Then, the extraction program deletes data points from the front and back of the speech list. This deletes extra noise data points that are part of the edge time slices. The deleting of data points is accomplished by comparing a moving average against an energy threshold. The energy threshold is calculated by equation 2.

$$Threshold = \mu_{speech\_list_i} + \beta * \sigma_{speech\_list_i} \quad i = 1, 2, \ldots, N \qquad (2)$$

where

N is the number of data points in a time slice

μ is the mean of the absolute values of the first time slice in the speech list

β is a multiplication factor

σ is the variance of the absolute values of the first time slice in the speech list

When the moving average of the absolute value of the speech list exceeds this threshold, then no more data points are deleted and the remaining data points in the speech list are written to a file. The number of time slices for these operations is always referenced from the beginning of the list.

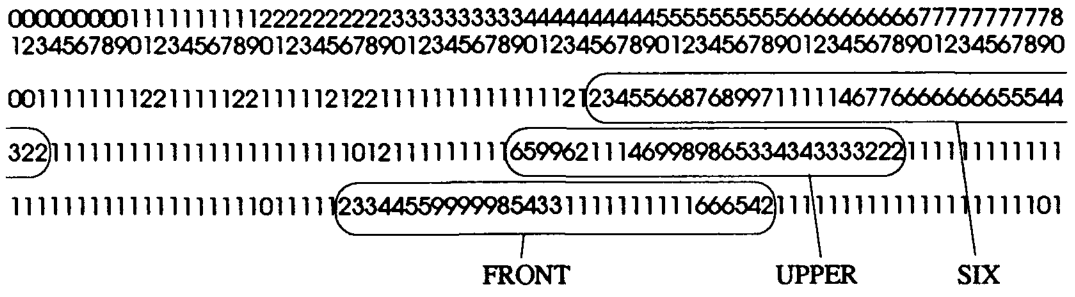
```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
00111111112211111221111121221111111111111121234556687689971111146776666666655544
3221111111111111111111111111012111111111165996211146998986533434333322211111111111
111111111111111111110111112334455999998543311111111116665421111111111111111111101
```

Figure 17 Speech characterization list

4.3 Database Management

The database manager manages the names of the words in the database list. When the ASRT starts up, the database manager reads the names of the words in the database from a file and stores them in a linked list. The database manager has two types of customers to manage, subprograms and users.

Many procedures and functions in ASRT need to access the database list to perform their operations. Sometimes two different procedures access this list in an alternating fashion. It is important that one procedure's access does not interfere with another's. To prevent this, the database manager allocates a pointer to any procedure requesting access to the list. Then, as a procedure uses the GET_NAME procedure, the database manager just increments the pointer for that procedure. The database manager returns an under_flow exception when a procedure tries to go past the end of the list. The procedures expect and use this exception to exit from their operating loops.

The user or a script can use the database manager to add and delete word names from the database list. Other functions include: displaying the words in the database list, setting the directory path for the database, setting a library name prefix, and toggling between using the main database list and database sublist. The capability to set a prefix allows the user to change the set of words or templates being used in the library for a given recognition test.

The database manager also manages the use of the sublist. Since a sublist is just a shorter version of the database list, the database manager can use all of the same functions to manage them. The difference is that the database manger uses a different set of pointers. The user or script file calls the toggle procedure to tell the database manger which set of pointers to use. The toggling between lists is transparent to the other functions and procedures in ASRT since all subprograms are database list length independent.

4.4 Features Operations

The feature_package contains all of the procedures and functions necessary to create the features used for recognition in ASRT. ASRT creates eight different features of each word. The creation of these features is computationally intensive and some features are derived from other features. To make creating the features as efficient as possible, all of the features are created at the same time. This reduces disk access and takes advantage of feature interdependencies. The following is the list of features:

1) Linear Prediction Power Spectrum (LPPS),

2) Delta LPPS (DPPS),

3) Linear Frequency Cepstrum Coefficients (LFCC),

4) Delta LFCC (DFCC),

5) Mel Frequency Mel Cepstrum (MFMC),

6) Delta MFMC (DFMC),

7) Bi-Linear Mel Cepstrum (BLMC),

8) Delta BLMC (DLMC).

4.4.1 LPPS and DPPS

The LPPS feature, as the name implies, is derived from the power spectrum. There exist many methods to calculate a power spectrum. This thesis uses the all poles - maximum entropy method because it is very good at fitting sharp spectral features (9:431). Equation 3 shows the relationship between the power spectrum and the α coefficients (9:431).

$$P(f) = \frac{\alpha_0}{\left| 1 + \sum_{k=1}^{M} \alpha_k z^k \right|^2} \tag{3}$$

where

M is the number of poles

α_k are coefficients derived from the data

The α_k 's are obtained by solving the matrix equation show in equation 4 (9:432).

$$\begin{bmatrix} \Phi_0 & \Phi_1 & \Phi_2 & \cdots & \Phi_M \\ \Phi_1 & \Phi_0 & \Phi_1 & \cdots & \Phi_{M-1} \\ \Phi_2 & \Phi_1 & \Phi_0 & \cdots & \Phi_{M-2} \\ \cdots & & & & \cdots \\ \Phi_M & \Phi_{M-1} & \Phi_{M-2} & \cdots & \Phi_0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{4}$$

The Φ's are obtain by equation 5 (9:431).

$$\Phi_j = \Phi_{-j} = \frac{1}{N+1-j} \sum_{i=0}^{N-j} c_i c_{i+j} \quad J = 0, 1, 2, \ldots, N \tag{5}$$

where

N is the number of data points

c_i 's are the sample data points

N for this thesis is 512 which results in a 256 point power spectrum. Energy is then added to the higher frequencies of the power spectrum by the amount of 6db per octave above 600 hertz. Finally, the 256 point power spectrum is reduced to 16 points by using a Mel frequency compression scheme. Figure 18 shows a mapping from the Mel frequency scale to the linear frequency scale. The Mel scale is divided into 16 equally spaced frequency bins. The bins are then mapped to the linear scale, also shown in Figure 18. All

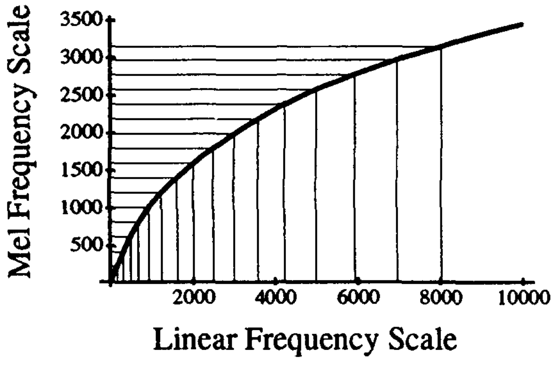

Figure 18 Linear scale to Mel scale mapping

amplitude values of the 256 point power spectrum falling within the same frequency bin are summed together, producing 16 values for the feature. Figure 19 shows a contour plot of

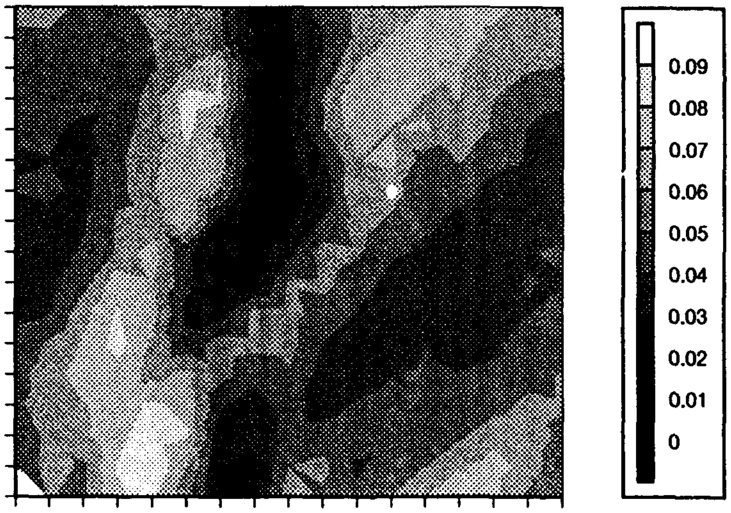

Figure 19 LPPS feature for the word six

LPPS for the word six. Time runs left to right and the higher frequencies are displayed at the bottom of the image. The high frequencies for the fricatives at the beginning and the end of the word six is shown by the lighter areas along the bottom. The image also shows a vertical valley where the energy drops off at the beginning of the X sound. The DPPS feature is the difference between two LPPS time slices and is defined by equation 6 (10:66,.

$$DPPS_i = DPPS_{i+1} - DPPS_{i-1}, \quad i = 2, 3, \ldots, N-1 \tag{6}$$

where

N is the number of time slices in the word.

Figure 20 shows a contour plot of DPPS for the word six. This image shows the

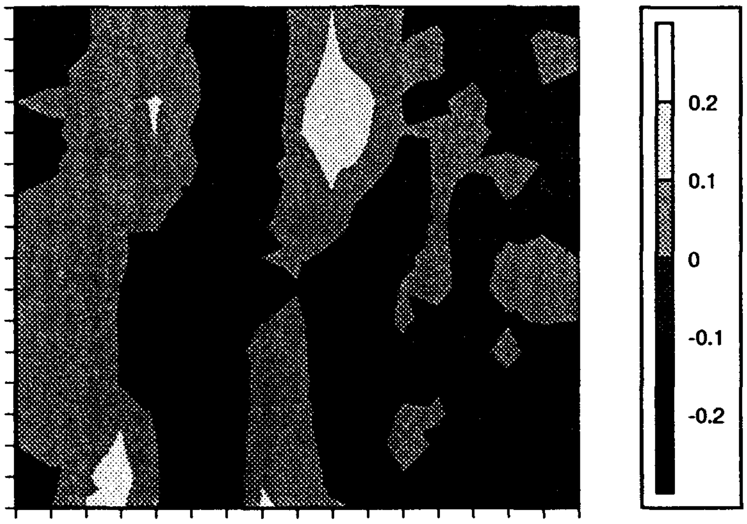

Figure 20 DPPS feature for the word six

beginning fricative very well but the ending fricative gets washed out. The X sound is accentuated, and the valley shown very clearly.

4.4.2 LFCC and DFCC

The LFCC feature is derived from the linear prediction power spectrum. This is the same power spectrum used for LPPS, but without the Mel compression scheme. Equation 7 converts the power spectrum into Cepstrum coefficients (11:67).

$$LFCC_i = \sum_{k=0}^{K-1} Y_k \cos\left(\frac{\pi i k}{K}\right), \quad i = 1, 2, \ldots, M, \tag{7}$$

where

Y is the power spectrum coefficients

K is the number coefficients

M is the number of desired Cepstrum coefficients

Figure 21 shows a contour plot of LFCC for the word six. This image is showing

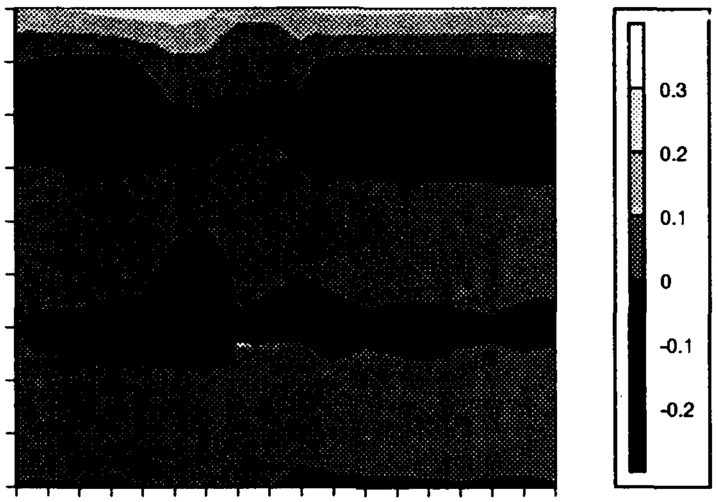

Figure 21 LFCC feature of the word six

the vocal tract model during the utterance of the word. This image is very smooth because the vocal tract changes relatively slowly while speaking. The X sound can be seen as the sharper movement during the middle time slices. The lighter regions along the top show the build up in the beginning fricative and the trail off of the ending fricative. The horizontal bands are the resonant frequencies of the vocal tract. The DFCC feature is the difference between two LFCC time slices and is defined by equation 8 (10:66).

$$DFCC_i = DFCC_{i+1} - DFCC_{i-1}, \quad i = 2, 3, \ldots, N-1 \tag{8}$$

where

N is the number of time slices in the word

Figure 22 shows a contour plot of DFCC for the word six. This image has sharper features in middle where the X sound is, but overall is fairly tame.

4.4.3 MFMC and DFMC

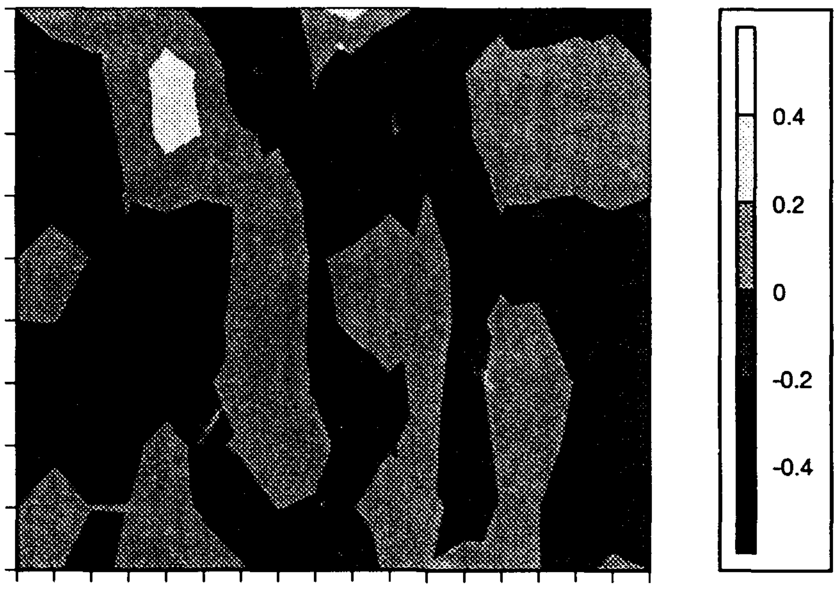

Figure 22 DFCC feature for the word six

The MFMC feature also uses the linear prediction power spectrum, but first warps the linear scale frequencies to Mel scale frequencies. The warping is done by simulating overlapping band pass filters. The filters are laid out in 16 Mel scale frequency bins, as shown in Figure 23. The 256 data points of the power spectrum arr positioned across the filters. All data points within a filter are multiplied by the corresponding weight value and

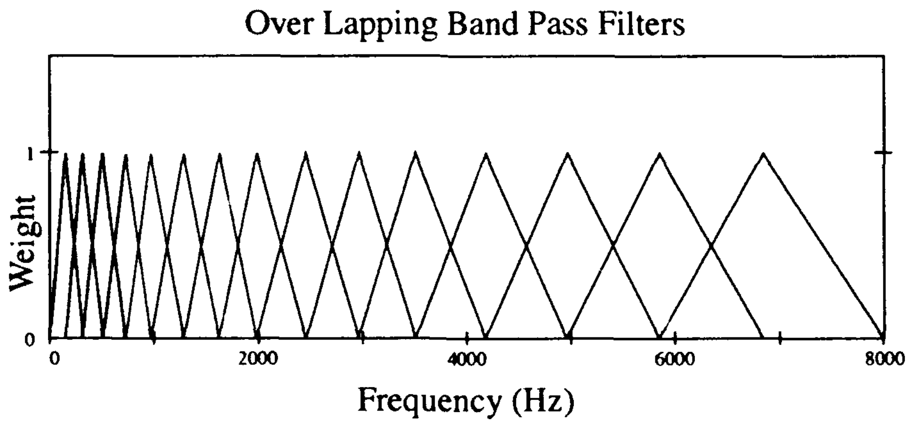

Figure 23 Overlapping band pass filters (11:67)

summed together and logged. Then, equation 9 is used to calculate the Mel cepstrum coefficients (11:67).

$$MFMC_i = \sum_{k=1}^{K} X_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right], \quad i = 1, 2, \ldots, M, \tag{9}$$

where,

X is the log energy values of the filter outputs

K is the number of overlapping bandpass filters

M is the desired number of Mel Cepstrum coefficients

Figure 24 shows a contour plot of MFMC for the word six. This image is also a

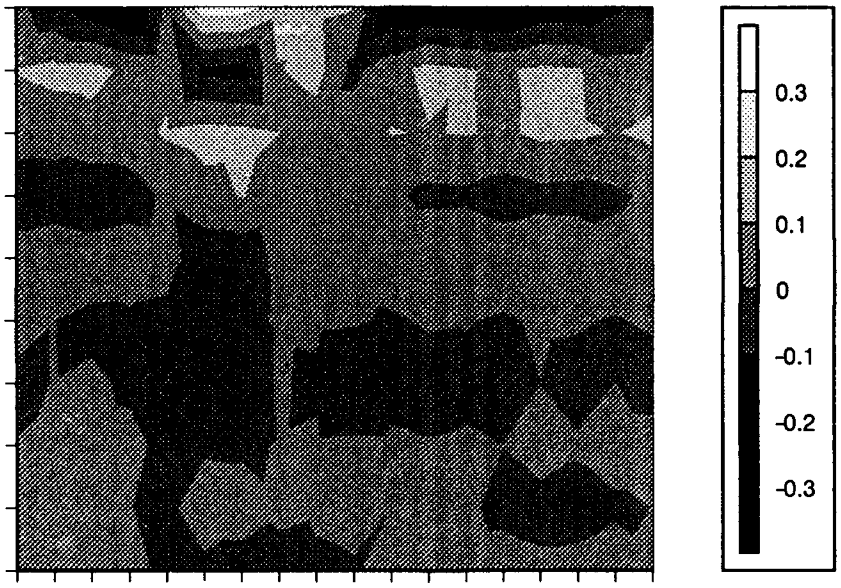

Figure 24 MFMC feature for the word six

model of the vocal tract. The dark regions along the top correspond to the fricatives at the beginning and end of the word. The plosive sound of the X is distinct in the middle of the of the image. This image shows remnants of the vocal tract resonant frequencies. The horizontal bands are now very distinct, probable because of the crudeness of the processing. The DFMC feature is the difference between two MFMC time slices and is defined by equation 10 (10:66).

$$DFMC_i = DFMC_{i+1} - DFMC_{i-1}, \quad i = 2, 3, \ldots, N-1 \tag{9}$$

where

N is the number of time slices in the word

Figure 25 shows a contour plot of DFMC for the word six.

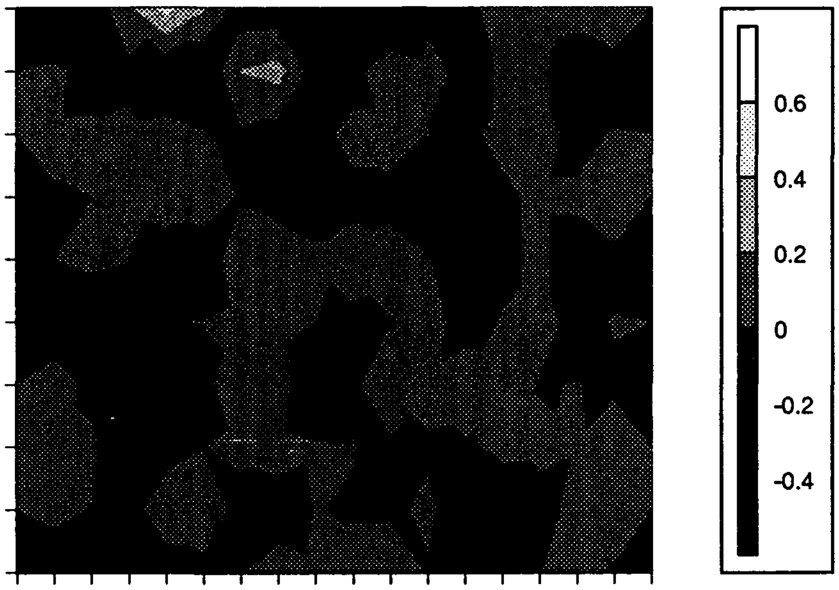

Figure 25 DFMC feature of the word six

4.4.4 BLMC and DLMC

The BLMC feature is derived directly from the LFCC feature. A bi-linear transform is used to warp the LFCC from a linear scale to Mel scale to create the new feature. The bi-linear transform is shown in equations 11 and 12 (10:64-65).

$$z_{new}^{-1} = \frac{(z^{-1} - \alpha)}{(1 - \alpha z^{-1})}, \quad (-1 < \alpha < 1) \tag{11}$$

$$\omega_{new} = \omega + 2 \tan^{-1}\left(\frac{\alpha \sin \omega}{1 - \alpha \cos \omega}\right) \tag{12}$$

where

ω is the sampling frequency

α is the warping parameter, 0.6 is used for this feature

Figure 26 shows a contour plot of BLMC for the word six. This image is directly

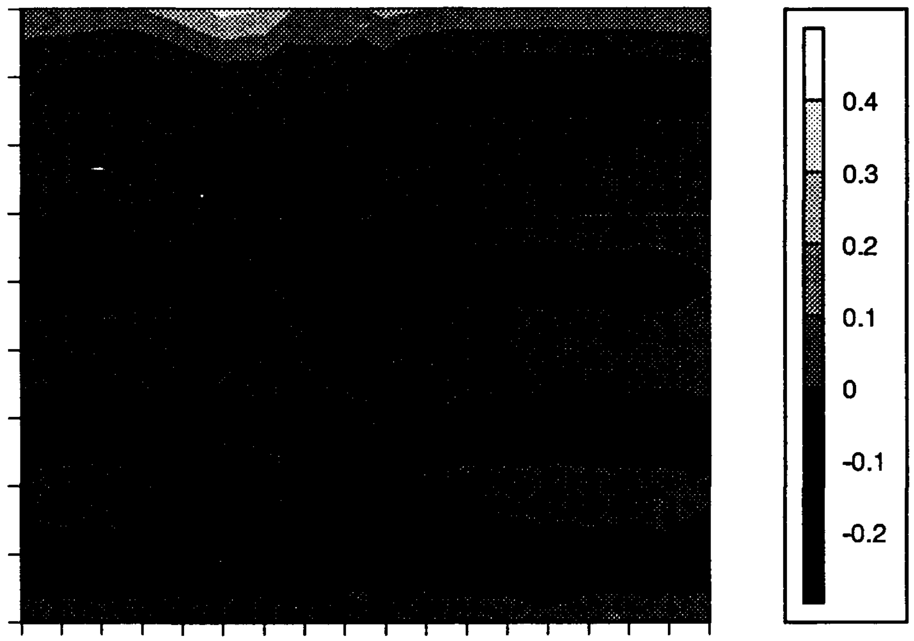

Figure 26 BLMC feature of the word six

comparable to the LFCC feature image and shows many of the same details. The major difference is the number of resonant frequencies shown, which may because of the frequency warp. The DLMC feature is the difference between two BLMC time slices and is defined by equation 13 (10:66).

$$DFMC_i = DFMC_{i+1} - DFMC_{i-1}, \quad i = 2, 3, \ldots, N-1 \tag{13}$$

where

N is the number of time slices in the word

Figure 27 shows a contour plot of DLMC for the word six.

All features are word normalized according to equations 14 and 15, shown here for the feature LPPS.

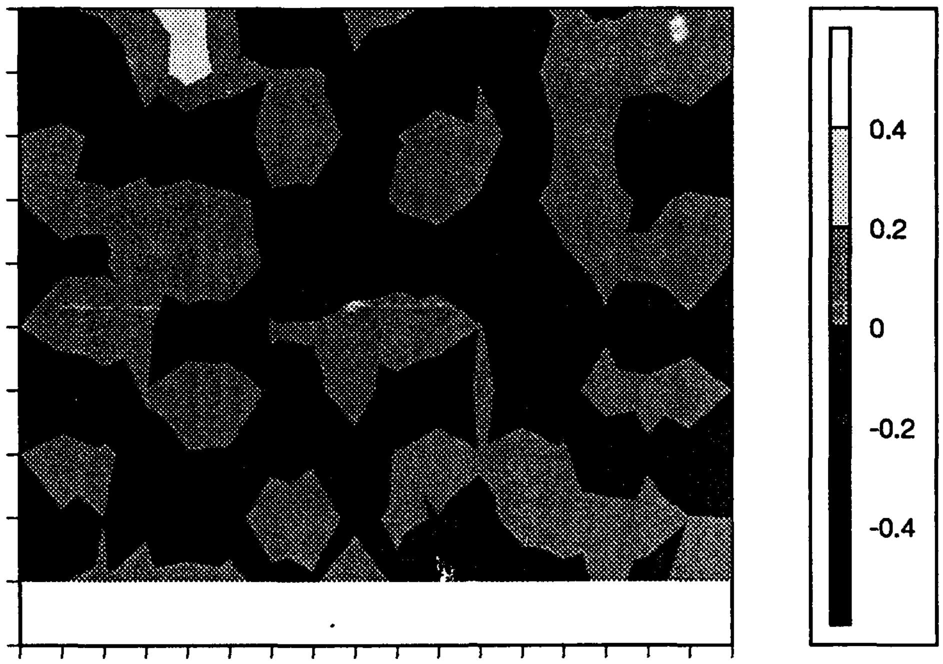

Figure 27 DLMC feature of the word six

$$NORM = \left[\sum_{i=1}^{N} \sum_{j=1}^{M} \left(LPPS_{ij} \right)^2 \right]^{\frac{1}{2}} \quad (14)$$

$$LPPS_{ij} = \frac{LPPS_{ij}}{NORM}, \quad i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, M \quad (15)$$

where

N is the number of time slices

M is the number of components in each time slice

This normalization technique sets the energy of each word to unity. Time slice normalization is not used because of the possibility that it may raise the energy levels of a noise time slice in comparison to the rest of the time slices in the word.

4.5 Recognition Operations

The RECOGNITION_PACKAGE contains all of the procedures and functions necessary to perform a recognition test between an unknown word and a list of words. The ASRT system uses three word recognizers. The recognizers are Dynamic Time Warping (DTW), Feature Image Recognition (FIR), and Error Surface Image Recognition (ESIR). All recognizers can use all of the speech features described in section 4.4 for recognition. Each feature must be brought online before it can be used. That is, the feature for each word in the library is read from the database and stored in that feature's linked list. This helps to improve recognition performance by eliminating a tremendous amount of disk accesses.

4.5.1 Dynamic Time Warping

The DTW algorithm tries to determine the best warping, aligning time slices, so that two words match the best. This algorithm is described in Chapter 2. One can better appreciate the task of the DTW algorithm by looking at actual error surfaces between words that are the same and words that are different. Figure 28 shows the error surface between the

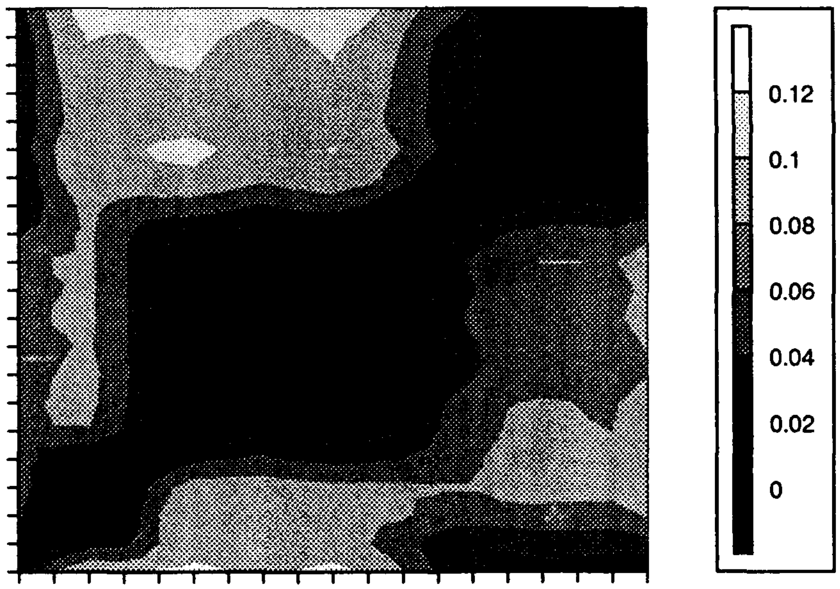

Figure 28 Error surface for simalar words

same word spoken twice by the same person. The dark region along the main diagonal has the smallest error values which indicates correlation between the two words. The DTW algorithm traverses the error surface by starting at the lower right hand corner and jumping one time slice at a time until a stopping condition is met. At each time slice the DTW algorithm looks at three possible jump locations and chooses the one with the smallest error value. The DTW distance is the sum of the error points along this path. Figure 29 shows the error surface between different words from the same person. There is no dark band of correlation, and the DTW distance value will be relatively large.

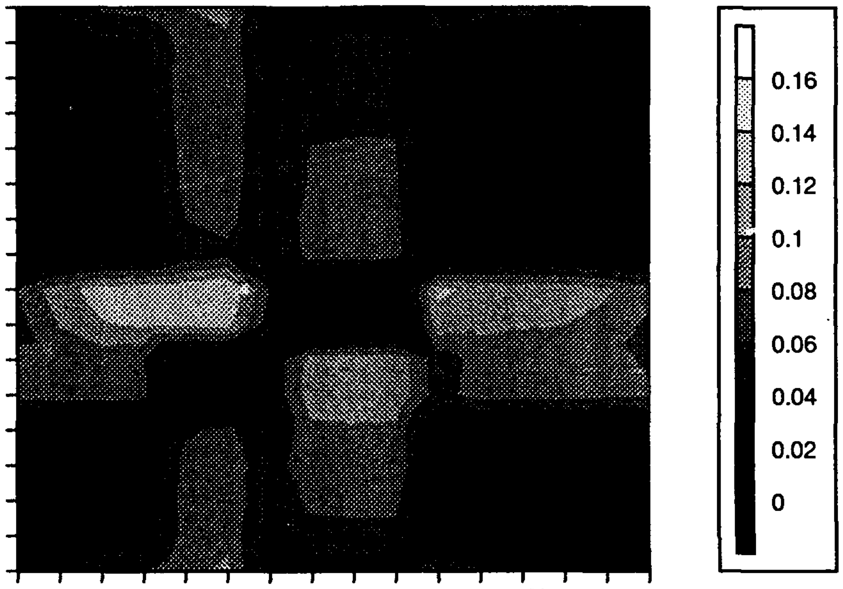

Figure 29 Error surface for different words

This implementation of DTW uses the Itakura rules for movement (2). The starting point of the path is constrained to the first time slice in each word. The end point, however, is not always constrained. If the DTW path runs into the ceiling or top of the error surface, then it follows along the ceiling to the upper right hand corner. In this case, the end points are constrained. If the DTW path runs into the right wall, then the algorithm stops, and the end points are not constrained. In this type of DTW implementation, the number of error points summed for each distance value is a function of the length of the unknown word.

4.5.2 Feature Image Recognition

The idea behind the FIR algorithm is to try to recognize a word by the image of its feature. The contour plots of the various features in Section 3 are essentially images. Figure 30 shows two contour plots of the LPPS feature for the word six said by the same talker.

The similarities are evident by visual inspection. Therefore, the low frequency values of a two-dimensional FFT should also be similar.

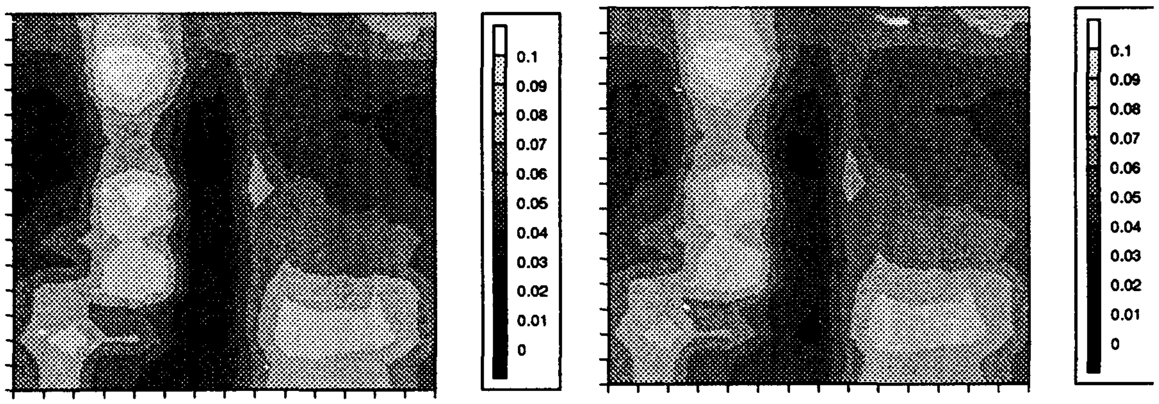

Figure 30 LPPC feature for two different sixes

One problem in speech is the varying length at which people say words. To overcome this problem in the FIR algorithm, a time warp alignment algorithm is used. This algorithm uses the DTW algorithm to determine the best match between the time slices in the two words. All words in the database are time warp aligned to best match the unknown or test word. Then, each word is run through a two dimensional Low Frequency (LF) FFT algorithm that scales the results to an eleven by eleven matrix (12:4-9). Figure 31 shows the LF matrix arrangement. Finally, each of the library word's LF matrices are compared to the unknown word's LF matrix. The comparison is done with a two dimensional Euclidean distance algorithm, which produces a single distance value.

4.5.3 Error Surface Image Recognition

The error surfaces shown in section 4.5.1 are also images. Therefore, they too can be described using the two dimensional FFT algorithm. The problem is what to compare them to for recognition purposes. A dark band running along the main diagonal, from the lower left to the upper right, signifies correlation in an error surface image. Thus, these

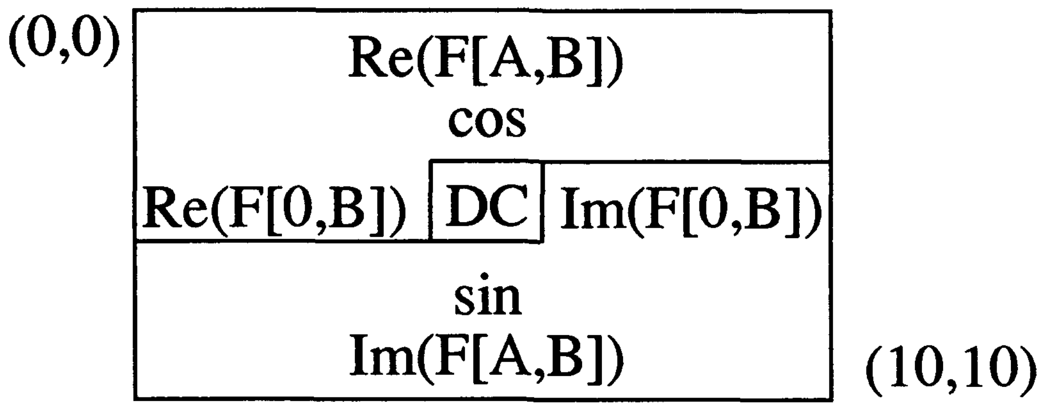

Figure 31 Layout of low frequency matrix

error surfaces can be compared to an error surface that is known to have this correlation. The ideal error surface for comparison is the one created between a word and itself. See Figure 32 .

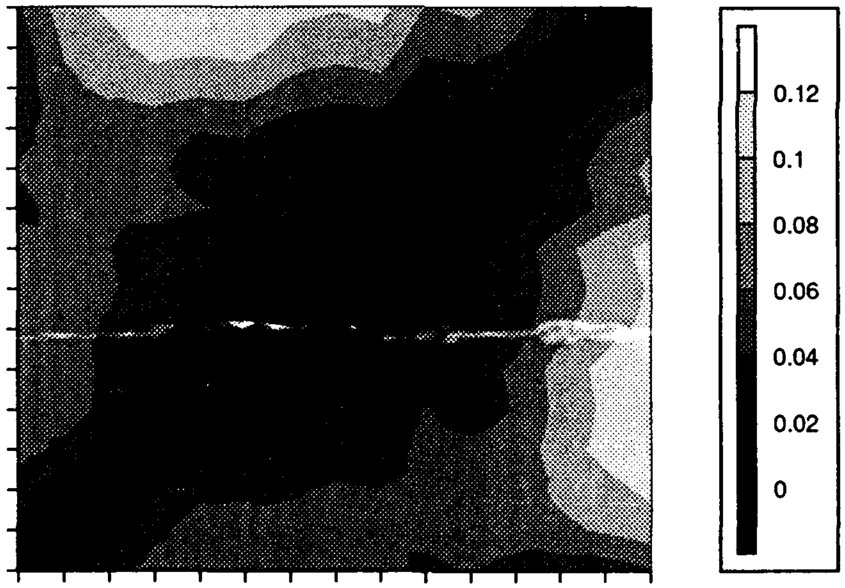

Figure 32 Perfect error surface

The ESIR algorithm uses the following steps: a referenced error surface is created between the the unknown word and itself. Then, the LF matrix is calculated using the two-dimensional LF FFT algorithm. Each word in the database list is then time warped aligned with the unknown word. The error surface is calculated between the unknown word and the new versions of the library words. The LF matrix is calculated for each error surface. Then, the Euclidean distance is calculated between the referenced LF matrix and the other LF matrices. These distance values are returned as the recognition results.

4.5.4 Hierarchical Recognition

The FIR and ESIR recognizers both use a two dimensional FFT algorithm and their execution times are much greater than for the DTW algorithm. Therefore, a hierarchical approach may be used to save processing time without losing recognition accuracy. The DTW recognizer is used first on all of the words in the database. Then, a new database sublist is created by selecting the top X number of words according to their distances. The other two recognizers are then run on the sublist of words and the winner is selected by feature fusion.

It is possible to make the second part of the hierarchical recognition conditional on the first part. An AGREE function is used to determine if the results of the first recognition pass agree. If they agree, then the recognition results stand. If the recognizers do not agree, then they create a sublist and go on to the second recognition test.

4.5.5 Fusion Recognition

The recognition package is limited to the eight features listed in Section 4.4 for recognition. However, the recognition package has the capability to combine features into super features and use them for recognition. The idea is to let the recognizer perform the fusion in the feature space, rather than in the recognition results space (13).

To use the fusion recognition, the user specifies the features to combine, the number of components in each, and a name for the new feature. Once specified, the recognition package creates the new feature of the library words from the feature lists that are already in memory. The new feature for the unknown word is created just before recognition, as the files are read from the database. All of the procedures and functions that are used for recognition are instantiated for specific features. Since a new feature is created, a special declaration section is needed to instantiate the procedures and functions to work on the number of components in the new feature. The user has great flexibility in creating new features with this set up, but the recognition speed is much slower as a result.

4.6 Fusion Operations

The FUSION_PACKAGE contains all of the procedures and functions necessary to integrate and display the results from the recognizers. The recognizers calculate distance values from the unknown word to all the library words. These distances are stored in a linked list and passed to the FUSION_PACKAGE. The FUSION_PACKAGE adds a label and stores all these results in another linked list. Figure 33 shows what this list of results looks like.

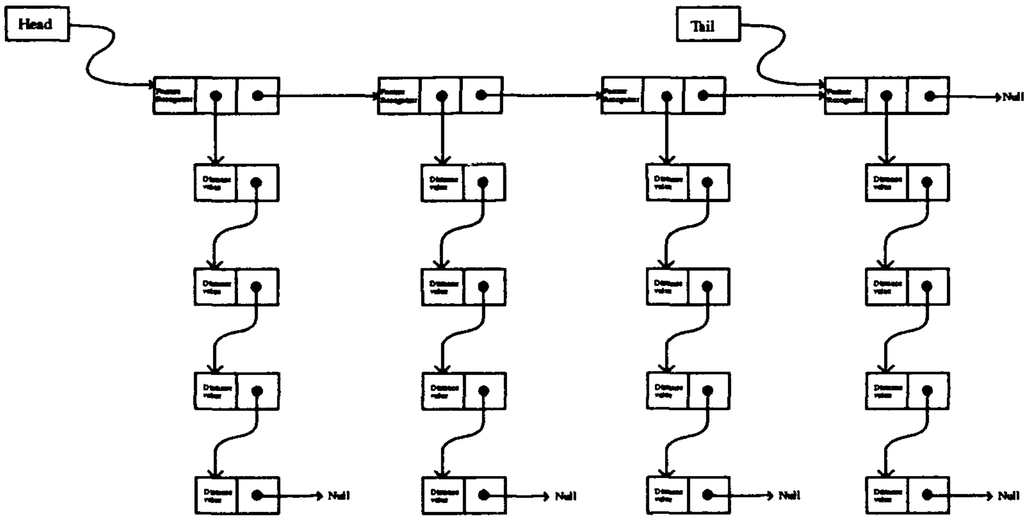

Figure 33 List structure for fusion operations

All of the procedures and functions in the fusion package are designed to work independently of what features are in the fusion list and how many distance values are in the results list. This is because the number of features used on any one recognition attempt is up to the user.

The basic fusion technique is to multiply the distance results for each feature by that feature's fusion factor, and then to sum all of the results corresponding to the same word. This fusion technique is described by equation 16.

$$Fusion\_Value_{word} = \frac{\sum_{i=1}^{K} Feature\_Result_{word_i} * Fusion\_Factor_i}{\sum_{i=1}^{K} Fusion\_Factor_i} \quad (16)$$

where

K is the number of recognition results in the fusion list

The fusion factor is normally 1.0, but the user can change this factor depending on the goodness of a particular feature. The higher the number the more influential the feature. The FUSION_PACKAGE can also display or write to a file the various results from the recognizers. The fusion list must be cleared before attempting to recognize the next word.

4.7 Template Operations

The TEMPLATE_PACKAGE contains all of the procedures and functions necessary to create and update templates for recognition. A template is a word created from multiple utterances of the same word. Templates help to improve recognition by being more representative of the variations in spoken words.

4.7.1 Creating Templates

The scheme used for creating templates is time warp alignment. The algorithm works by first determining which template component consists of the most time slices. The other template components are then time warp aligned to the longest word. Then, all corresponding time slices are summed together and averaged. This newly created word is the template, and is stored in the recognition database. Figure 34 shows two utterances of the word six and a template version.

4.7.2 Updating Templates

Templates may also be updated. As new talkers use the system, it is more convenient to update a template than to completely recreate the templates from scratch. The update

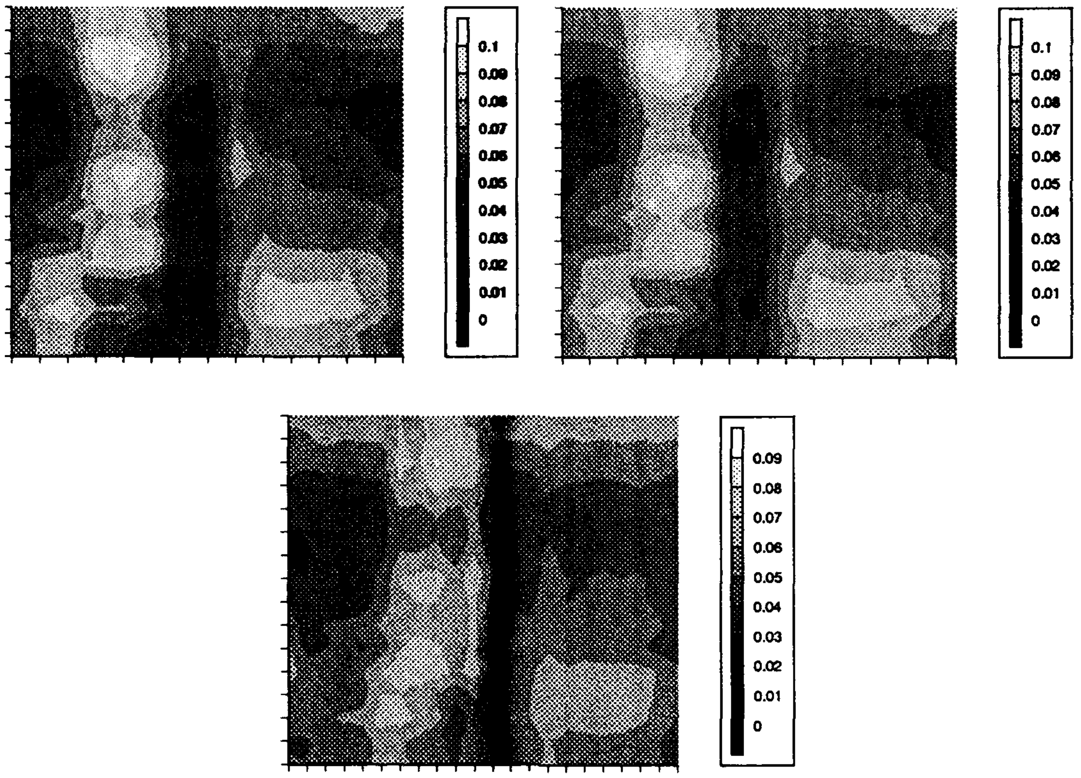

Figure 34 Template from two components

algorithm combines the template word with the new word to update the template. The update algorithm uses a template factor to direct the tracking of the update of the template. Tracking refers to how the template reflects the population of talkers. A fast tracking system will quickly move the template to reflect the latest talker. A slow tracking system better reflects the population of talkers that make up the template. The update algorithm uses equation 17 which incorporates a variable tracking speed.

$$Template\_new_{ij} = \frac{template\_old_{ij} * template\_factor + new\_word_{ij}}{1 + template\_factor} \tag{17}$$

where

i is the index for the number of components

j is the index for the number of time slices

A low template factor value is used for fast tracking. A value of one would average the template with the new word. A template factor equal to the number of words in the template would equally represent all words in the template.

4.9 Script Interpreter

The script interpreter is used to execute commands in a script file. Scripts allow the user to design a test and execute it without the necessity of going through the menus of the subsystems. Also, scripts make repeating a test convenient since the commands are already saved in a file.

The script interpreter is not a separate package. It is accessed from the main menu, but executes commands directly from any of the subsystems. The user produces a script by organizing commands in a script file. A script file is much like any other software program, in that the user sets up certain parameters, executes repetitive commands in a loop, makes decisions, displays results, and then exits. Each script may have many loops and if-then-else statements. Each loop is designed to increment through the words in the database list or sublist or through unknown file names. The loops exit after the last word in the list is accessed. Figure 35 shows a typical script file. A complete list of the the script commands and how to create them is in the users manual, Appendix A.

4.3 Summary

This chapter discussed in detail the design of the ASRT system. A users manual is included in Appendix A. The next chapter shows the test results using ASRT.

```
SET_FILE_DIRECTION
AUTO_CREATE
SET_PREFIX
tom4_
CREATE_REPORT_FILE
TW_RESULTS.DAT
SET_DIRECTORY_NAME
[TRATHBUN.SPEECH.DATABASE]
SET_LIBRARY_PREFIX
TOM_
- load library
UPDATE_WORD_LIST
LPPS
SET_RECOGNIZER
TIME_WARP
- start loop
LOOP
RECOGNIZE
- get word from database list
NEXT_WORD
THIS_WORD
- recognize word using feature
RECOGNIZE
LPPS
FUSION
- output results
ALL_FEATURE_SMALLEST_RESULT
CLEAR_FUSION_LIST
- goto start of loop
GOTO
RECOGNIZE
CLOSE_OUTPUT_FILE
END
```

Figure 35 Typical script file

V Test Results

5.1 Introduction

This chapter presents the results from testing the ASRT system. ASRT is capable of performing numerous tests. However, the most rigerous tests will be run only for the talker dependent tests. Only the most promising recognition tests will be run for the talker independent tests. The talker for the talker dependent tests recorded three sets of the test words. The talkers for the talker independent tests recorded two sets of the test words. The test words are the digits zero through nine and the letters A through Z.

5.2 Talker Dependent

The three recognizers will first be tested separately, and then in combination. The combined tests will use sublists in a two step recognition process.

5.2.1 Recognizer Performance Tests

Tables 1 through 9 show the results for the DTW recognizer using all eight features. Tables 10 through 18 show the results for the FIR recognizer using all eight features. Tables 19 through 27 show the results for the ESIR recognizer using all eight features. The numbers and words along the left of each table are the correct results. Along the top of the table are the feature name columns, followed by the fusion column. These columns show the recognition results for each test. The cells in white contain incorrect results, while the grayed cells contain correct results. The total number wrong are counted along the bottom of the table. The nine test runs include six for testing the combinations of the three recorded word sets and three for testing with the templates. All features were given equal weight for the fusion result.

The DTW recognizer performed slightly better than the FIR recognizer, and both out performed the ESIR recognizer based on fusion results. Accuracy for DTW and FIR ranged from 86.1% to 97.2% correct. The accuracies for ESIR ranged from 75.0% to

TABLE 1 DTW RESULTS FOR TOM2 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	V	T	V	P	D	P	T	T
C	C	C	C	C	C	C	C	C	C
D	D	V	D	D	D	B	D	D	D
E	E	E	E	E	E	E	E	V	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	7	J	J	J	K	J
K	K	K	K	J	K	K	K	J	K
L	L	L	L	2	L	L	L	L	L
M	M	M	M	M	M	M	M	M	M
N	M	M	N	M	N	N	M	N	N
O	O	O	O	O	O	O	O	O	O
P	T	K	T	D	P	D	P	T	T
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	I	R	I	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	5	T	T	T	T	T
U	2	Q	U	2	U	U	U	2	U
V	E	V	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	I	Y	Y	Y	I	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	5	6	2	9	1	3	2	7	2

TABLE 2 DTW RESULTS FOR TOM3 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	E	4	4	4	4	4
5	5	K	5	7	5	7	5	7	7
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	K	A	K	A	A	A	K	A
B	D	B	B	B	B	D	B	A	B
C	C	C	C	C	C	C	C	C	C
D	D	E	D	E	D	D	D	E	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	T	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	K	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	E	L	E	L	L	L	E	L
M	M	M	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	E	O	O	O	O	O
P	P	T	P	T	T	T	T	T	T
Q	W	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	Q	U	U	U	U	U	U	U
V	D	E	D	E	D	V	D	E	D
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	3	8	2	9	2	3	2	8	3

TABLE 3 DTW RESULTS FOR TOM1 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	M	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	D	A	E	E	D	E	E	E	E
C	C	C	C	C	C	C	C	C	C
D	D	D	D	T	D	B	D	D	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	0	J	J	J	J	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	N	M	N	M	M	M	N	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	P	B	P	B	P	B	P	B	P
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	M	S	S	S	S	S	S	S
T	T	T	T	T	T	G	T	G	T
U	U	U	U	U	U	U	U	U	U
V	V	E	V	E	E	V	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	2	4	3	4	2	4	2	4	1

TABLE 4 DTW RESULTS FOR TOM3 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	I	I	5	7	5	J	5	7	I
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	H	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	8	A	A	A	A	A
B	T	K	D	K	D	K	D	K	T
C	C	C	C	C	C	C	C	C	C
D	D	V	D	T	D	E	D	D	D
E	E	E	E	E	E	E	E	E	E
F	K	F	L	F	F	F	L	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J	J	J	J	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	E	L	L	L	L	L
M	N	M	N	M	M	M	N	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	D	T	D	T	D	E	D	T	T
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	G	T	T	G	T	G	T	G	T
U	U	U	U	U	U	U	U	U	U
V	D	D	D	V	D	V	D	V	D
W	W	W	W	X	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	I	I	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	8	6	5	8	3	5	5	4	4

TABLE 5 DTW RESULTS FOR TOM1 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	Y	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	D	A	D	A	A	A	D	A
B	B	D	B	D	P	E	B	D	D
C	C	C	C	C	C	C	C	C	C
D	V	V	V	T	V	T	V	T	V
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	K	K	K	J	J	J	K	K	K
K	K	K	K	K	K	K	K	K	K
L	L	O	L	E	L	F	L	O	L
M	M	M	M	M	M	M	M	M	M
N	N	M	N	M	N	M	N	M	M
O	O	O	O	O	O	O	O	O	O
P	P	K	P	K	P	T	P	7	P
Q	2	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	P	A	P	E	P	T	P	A	P
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	5	8	3	8	3	6	3	8	5

TABLE 6 DTW RESULTS FOR TOM2 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	I	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	D	E	E	V	T	V	E	V
C	C	C	C	C	C	C	C	C	C
D	V	D	V	D	V	D	D	D	D
E	E	E	E	E	E	E	E	E	E
F	X	F	F	F	F	S	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	Y	I	I
J	J	J	J	J	J	J	J	J	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	A	L	L	L	L	L
M	N	M	N	M	M	M	M	M	M
N	N	M	N	M	N	M	N	M	N
O	O	O	O	O	O	O	O	O	O
P	V	T	V	T	P	T	T	T	T
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	D	T	D	T	T	T	D	T
U	U	U	U	U	U	U	U	U	U
V	E	E	V	V	V	E	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	I	Y	9	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	6	6	4	7	2	5	3	5	2

TABLE 7 DTW RESULTS FOR TOM3 VS TOM12

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	I	I	5	7	5	7	5	7	I
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	K	A	E	A	A	A	K	A
B	D	B	B	B	B	D	B	A	B
C	C	C	C	C	C	C	C	C	C
D	D	T	D	T	D	D	D	T	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	S	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	K	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	N	M	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	D	T	T	T	D	T	T	T	T
Q	W	Q	2	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	Q	U	U	U	U	U	U	U
V	D	E	D	E	D	V	D	E	D
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	6	7	4	6	2	4	2	8	3

TABLE 8 DTW RESULTS FOR TOM2 VS TOM13

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	P	V	V	D	D	P	P	P
C	C	C	C	C	C	C	C	C	C
D	D	V	D	D	D	D	D	D	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	C	J	J	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	M	M	M	M	M	M	M	M	M
N	M	N	N	N	N	N	N	N	N
O	O	O	O	E	O	B	O	O	O
P	P	P	P	D	P	D	P	D	P
Q	Q	Q	Q	2	Q	2	Q	2	Q
R	R	I	R	N	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	D	T	D	T	D	T
U	Q	2	U	2	U	U	U	2	U
V	V	E	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	1	Y	1	Y	Y	Y	1	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	2	6	1	9	1	5	1	7	1

TABLE 9 DTW RESULTS FOR TOM1 VS TOM23

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	D	D	E	D	E	E	E	D	D
C	C	C	C	C	C	C	C	C	C
D	D	B	D	D	D	P	D	O	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J	J	K	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	E	L	L	L	L	L
M	N	N	N	N	M	M	N	M	N
N	N	N	N	N	N	N	N	N	N
O	O	O	O	E	O	O	O	O	O
P	P	7	P	B	P	D	P	B	P
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	P	T	G	T	G	T	T	T
U	U	U	U	U	U	U	U	U	U
V	V	E	V	E	E	E	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	2	6	2	7	2	5	2	6	2

TABLE 10 FIR RESULTS FOR TOM2 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	Y	I	I	9	I	I	I	I	I
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	V	E	V	P	D	P	T	V
C	C	C	C	C	C	C	C	C	C
D	D	E	D	E	D	B	D	D	D
E	E	E	E	E	E	V	E	V	E
F	F	F	F	F	F	F	F	F	F
G	G	G	T	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	2	L	L	L	L	L
M	M	N	M	M	M	M	M	M	M
N	M	N	M	N	N	N	M	N	N
O	O	O	O	O	O	L	O	O	O
P	T	K	T	D	P	P	P	T	T
Q	Q	Q	Q	Q	Q	Q	Q	2	Q
R	R	R	R	O	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	2	U	U	U	U	U	U	U
V	E	V	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	I	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	5	6	5	8	2	6	3	5	3

TABLE 11 FIR RESULTS FOR TOM3 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	Q	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	O	4	E	4	4	4	4	4
5	I	Y	5	7	5	5	I	I	7
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	V	A	H	A	A	A	A	A
B	D	A	B	A	B	D	B	A	B
C	C	C	C	C	C	C	C	C	C
D	D	E	D	B	D	D	D	E	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	T	G	T	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	K	J	K	J	J	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	E	L	L	L	E	L
M	M	N	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	P	E	T	T	P	T	T	T	T
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	D	E	D	V	D	E	D	V	E
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	4	9	3	8	1	4	3	6	3

TABLE 12 FIR RESULTS FOR TOM1 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	D	A	D	A	E	E	D	D	D
C	C	C	C	C	C	C	C	C	C
D	D	P	D	T	D	P	D	D	D
E	E	V	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	K	J	0	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	O	L	L	L	L	L
M	N	M	N	M	M	N	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	P	7	P	B	P	P	P	P	P
Q	Q	2	Q	Q	Q	2	Q	Q	Q
R	R	4	R	R	R	R	R	4	R
S	S	M	S	S	S	S	S	F	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	V	E	V	V	V	V	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	2	9	2	5	1	4	1	5	1

TABLE 13 FIR RESULTS FOR TOM3 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	I	I	5	7	5	5	I	7	I
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	W	7	7	7	7
8	8	H	8	H	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	K	A	8	A	A	A	A	A	A
B	D	K	D	K	E	K	D	K	D
C	C	C	C	C	C	C	C	C	C
D	D	T	D	T	D	E	D	T	D
E	E	E	E	E	E	E	E	V	E
F	K	F	S	S	F	S	S	F	S
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J	J	J	J	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	M	M	M	N	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	D	T	D	T	D	E	D	T	D
Q	Q	Q	Q	2	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	G	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	D	V	D	V	D	E	D	V	D
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	I	Y	1	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	6	6	5	10	4	4	5	5	5

TABLE 14 FIR RESULTS FOR TOM1 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	Y	1	1	1	1	1
2	2	2	2	2	Q	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	R	4	4	4	4	4	4
5	Y	Y	Y	5	I	5	Y	5	Y
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	3	D	A	D	A	A	3	E	A
B	B	D	B	D	V	E	B	E	D
C	C	C	C	C	C	C	C	C	C
D	D	V	D	D	P	T	D	D	V
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	K	K	K	J	J	J	J	J	J
K	K	K	K	K	K	K	K	K	K
L	L	O	L	O	L	L	L	O	L
M	M	M	M	M	M	M	M	M	M
N	M	M	M	M	N	N	N	M	M
O	O	O	O	O	O	O	O	O	O
P	P	K	P	K	P	T	P	T	V
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	F	S	F	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	T	V	F	V	P	V	P	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	5	9	5	7	5	3	3	5	5

TABLE 15 FIR RESULTS FOR TOM2 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	R	5	5	5	I	5	I	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	M	V	E	D	D	T	D	E
C	C	C	C	C	C	C	C	C	C
D	V	D	V	D	V	D	D	D	D
E	E	E	E	E	P	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	Y	I	I
J	J	J	J	J	J	J	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	O	L	L	L	L	L
M	N	M	M	M	M	M	M	M	M
N	N	M	N	M	N	N	N	M	N
O	O	O	O	O	O	O	O	O	O
P	P	T	T	T	P	T	P	P	T
Q	Q	2	Q	Q	Q	Q	Q	Q	Q
R	R	4	R	S	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	D	T	D	T	T	T	D	T
U	U	U	U	U	U	U	U	U	U
V	E	E	V	V	V	V	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	1	Y	Y	Y	Y	I	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	5	6	4	5	4	2	3	5	2

TABLE 16 FIR RESULTS FOR TOM3 VS TOM12

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	Q	2	2
3	3	3	3	3	3	3	3	3	3
4	4	R	4	E	4	4	4	4	4
5	I	Y	5	7	5	5	I	7	I
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	K	V	A	E	A	A	A	K	A
B	D	A	B	A	B	D	B	A	B
C	C	C	C	C	C	C	C	C	C
D	D	E	D	T	D	D	D	D	D
E	E	E	E	E	E	E	E	E	E
F	F	S	F	F	F	S	F	S	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	K	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	N	M	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	T	T	T	T	D	T	T	T	T
Q	2	Q	2	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	D	E	D	V	D	E	D	V	E
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	I	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	7	9	3	7	2	5	5	6	3

TABLE 17 FIR RESULTS FOR TOM2 VS TOM13

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	9	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	V	V	V	V	D	D	P	P	P
C	C	C	C	C	C	C	C	C	C
D	D	E	D	D	D	D	D	D	D
E	E	V	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	L	L	L	O	L	L	L	L	L
M	N	M	M	M	M	M	M	M	M
N	M	N	N	N	N	N	N	N	N
O	O	O	O	O	O	L	O	O	O
P	P	P	T	D	P	P	P	P	P
Q	Q	Q	Q	Q	Q	2	Q	2	Q
R	R	I	R	O	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	T	T	D	T	T	T	T	T
U	U	2	U	U	U	U	U	U	U
V	V	V	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	9	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	3	5	2	8	1	4	1	2	1

TABLE 18 FIR RESULTS FOR TOM1 VS TOM23

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	D	D	D	D	E	E	E	D	D
C	C	C	C	C	C	C	C	C	C
D	P	P	D	T	P	D	D	P	P
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J	K	K	K	J
K	K	K	K	K	K	K	K	J	K
L	L	O	L	O	L	L	L	L	L
M	N	N	N	N	M	M	M	M	N
N	N	N	N	N	N	N	N	N	N
O	O	O	O	E	O	O	O	E	O
P	P	V	P	B	P	B	P	P	P
Q	Q	2	Q	Q	Q	2	Q	Q	Q
R	R	R	R	R	R	R	R	4	R
S	S	S	S	S	S	S	S	S	S
T	T	P	T	P	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	V	E	V	V	V	V	V	E	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	6	Z
Wrong	3	7	2	6	2	4	2	7	2

TABLE 19 ESIR RESULTS FOR TOM2 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	Y	1	Y	I	1	1	Y	1	Y
2	Q	2	2	2	2	2	2	2	2
3	K	3	3	3	3	T	3	3	3
4	K	R	K	I	K	4	K	2	4
5	5	O	5	5	5	5	5	Y	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	H	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	3	A	A	A	V	A	A	A
B	E	V	V	V	P	D	T	V	V
C	C	C	C	C	C	C	C	C	C
D	D	O	B	D	D	D	D	D	D
E	E	E	E	V	D	V	P	V	E
F	F	F	F	F	F	S	F	F	F
G	G	G	C	G	C	G	C	G	G
H	8	H	H	H	H	H	H	H	H
I	I	K	Y	I	I	I	I	I	I
J	J	J	J	7	J	K	J	2	J
K	K	K	K	3	K	K	K	K	K
L	L	V	L	2	L	L	L	L	L
M	2	M	O	M	M	M	M	M	M
N	M	M	M	M	N	N	M	M	M
O	O	V	O	U	O	A	O	V	O
P	T	K	T	2	D	D	P	D	T
Q	Q	Q	Q	2	Q	C	Q	2	Q
R	R	V	4	K	R	Y	R	Y	R
S	S	S	S	S	S	S	S	S	S
T	T	2	T	T	T	T	T	T	T
U	Q	U	U	U	U	U	U	U	U
V	E	2	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	I	Y	I	Y	Y	I	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	12	13	10	12	4	11	5	10	4

TABLE 20 ESIR RESULTS FOR TOM3 VS TOM1

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	Y	1	Y	1	1	1	Y	1	1
2	Q	2	Q	2	Q	2	Q	2	Q
3	3	3	3	3	3	3	3	3	3
4	R	2	4	E	4	4	4	2	4
5	Y	2	7	Y	Y	7	Y	5	Y
6	6	6	6	6	6	6	6	6	6
7	N	7	2	7	K	7	J	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	1	9	9	9	I	9
A	K	V	O	E	A	A	3	N	K
B	D	4	B	3	B	B	B	A	B
C	C	C	C	C	C	C	C	C	C
D	D	2	2	T	D	D	D	E	D
E	E	E	E	E	E	E	E	E	E
F	1	F	F	F	F	F	S	F	F
G	T	G	J	0	C	T	C	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	7	J	K	J	K	J	J	J
K	K	K	K	K	K	K	K	K	K
L	M	E	M	E	L	L	L	E	L
M	M	N	M	N	M	M	N	M	M
N	M	N	M	M	N	N	O	N	M
O	O	E	O	E	O	A	O	A	O
P	2	E	2	T	P	P	T	T	T
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	2	T	T	T	G	T	T	T
U	U	U	U	U	U	U	U	U	U
V	B	E	B	E	B	V	B	O	E
W	W	W	W	W	W	W	W	W	W
X	X	8	X	8	X	X	X	X	X
Y	Y	I	Y	Y	Y	I	Y	I	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	13	14	11	14	5	6	11	10	6

TABLE 21 ESIR RESULTS FOR TOM1 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	M	1	M	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	A	3	3	3	3	K	3	3	3
4	R	4	R	K	R	4	R	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	A	A	A	A	A	A	A	A	A
B	D	A	D	A	D	E	D	D	D
C	C	C	C	C	C	C	C	C	C
D	T	D	T	T	E	D	D	D	D
E	E	V	E	V	E	V	E	V	E
F	F	F	F	F	F	F	F	F	F
G	G	0	E	0	C	G	G	0	C
H	X	H	X	H	H	H	H	H	H
I	I	Q	I	I	I	I	I	I	I
J	J	K	2	0	J	J	J	J	J
K	K	2	K	K	K	K	K	K	K
L	L	L	L	O	L	L	L	L	L
M	M	M	M	M	M	M	M	M	M
N	N	M	N	N	N	N	N	N	N
O	O	V	O	E	O	E	O	E	O
P	P	P	T	P	B	B	T	P	P
Q	Q	2	Q	U	Q	2	Q	2	Q
R	5	T	R	V	R	R	R	4	R
S	S	M	S	M	S	S	S	M	S
T	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	V	E	V	E	V	V	V	E	E
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	I	Y	I	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	7	13	8	13	5	5	3	8	3

TABLE 22 ESIR RESULTS FOR TOM3 VS TOM2

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	I	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	R	4	R	4	4	4	4	4	4
5	I	E	J	I	I	J	I	7	I
6	6	6	6	6	6	6	6	6	6
7	J	7	J	7	J	7	J	7	J
8	8	H	8	H	8	8	8	H	8
9	I	9	9	9	9	9	9	I	9
A	8	U	8	U	8	A	8	8	A
B	2	K	D	K	D	K	D	K	D
C	C	C	C	C	C	C	C	C	C
D	D	P	D	V	D	T	D	V	D
E	E	V	E	E	E	E	E	V	E
F	K	F	K	8	M	F	M	F	K
G	G	G	G	G	G	G	G	G	G
H	X	H	H	H	H	H	H	H	H
I	I	9	I	I	I	I	I	Y	I
J	J	G	J	J	J	J	J	7	J
K	K	2	K	K	K	K	K	K	K
L	L	L	L	L	L	L	L	L	L
M	J	M	N	M	N	N	N	M	N
N	M	N	N	N	N	N	N	N	N
O	O	O	O	O	O	O	O	O	O
P	D	P	T	T	E	E	T	V	T
Q	Q	Q	Q	2	Q	Q	Q	2	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	G	G	G	G	T	G	T	G	G
U	U	U	U	U	U	U	U	U	U
V	D	D	D	V	D	V	D	V	D
W	W	W	W	5	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	I	1	Y	1	Y	Y	Y	Y	1
Z	Z	3	Z	Z	Z	Z	Z	Z	Z
Wrong	14	13	10	11	8	6	8	12	9

TABLE 23 ESIR RESULTS FOR TOM1 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	M	1	1	1	1	1
2	2	2	2	T	K	T	K	2	2
3	3	3	3	3	3	3	3	3	3
4	R	4	R	4	4	4	R	4	4
5	O	A	D	A	I	Y	R	F	Y
6	6	6	6	6	6	6	6	6	6
7	J	7	2	7	7	7	7	7	7
8	X	8	8	H	8	8	8	8	8
9	I	9	I	9	9	9	9	9	9
A	3	D	3	D	A	E	3	D	D
B	D	D	D	D	V	E	B	E	D
C	C	C	C	C	C	C	C	C	C
D	B	0	D	D	D	D	V	D	D
E	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F	F	F	F	F
G	G	Z	G	T	C	T	G	T	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	K	K	2	0	7	K	K	J	K
K	K	K	K	K	K	K	K	K	K
L	V	O	L	U	L	F	L	U	L
M	M	M	M	M	M	M	M	M	M
N	M	M	M	M	N	M	M	M	M
O	O	D	O	U	O	E	O	8	O
P	P	K	K	E	P	T	P	E	E
Q	2	Q	2	Q	Q	Q	Q	Q	Q
R	R	I	R	R	R	R	R	R	R
S	S	F	S	F	S	S	S	F	S
T	T	D	T	T	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	E	M	P	M	P	E	P	E	E
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	A	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	13	15	11	13	6	11	8	9	7

TABLE 24 ESIR RESULTS FOR TOM2 VS TOM3

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	I	4	I	4	4	4	4	4	4
5	R	5	R	5	I	5	I	5	5
6	6	6	6	6	6	6	6	6	6
7	J	7	J	7	2	7	J	7	J
8	8	8	8	8	8	8	8	8	8
9	I	9	9	9	9	9	9	9	9
A	3	A	3	A	8	A	3	A	A
B	T	5	T	E	P	D	T	E	E
C	C	C	C	C	C	C	C	C	C
D	D	D	V	D	P	E	P	E	E
E	E	E	E	E	P	E	P	E	E
F	X	F	X	F	F	S	S	F	F
G	G	T	G	G	G	G	G	T	G
H	X	H	8	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	2	J	G	J	G	J	G	J
K	K	0	K	3	K	3	K	0	K
L	D	U	L	U	L	Q	L	U	L
M	S	M	N	M	M	M	M	M	M
N	N	M	N	M	N	M	N	M	N
O	O	O	O	5	O	O	O	O	O
P	T	T	T	T	P	T	D	T	T
Q	Q	2	2	Q	Q	Q	Q	2	Q
R	R	E	R	E	R	O	R	R	R
S	S	S	S	H	S	S	S	S	S
T	T	D	T	D	T	D	T	D	D
U	U	U	U	U	U	U	U	U	U
V	E	E	T	E	V	E	P	E	E
W	N	W	N	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	9	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	13	11	13	12	5	12	8	11	6

TABLE 25 ESIR RESULTS FOR TOM3 VS TOM12

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	I	1	1
2	Q	2	Q	2	Q	2	Q	2	Q
3	3	3	3	3	3	3	3	3	3
4	R	2	4	E	4	4	4	2	R
5	I	2	7	Y	I	7	I	7	I
6	6	6	6	6	6	6	6	6	6
7	N	7	2	7	J	7	J	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	I	I	9	9	9	I	9
A	M	V	O	E	A	A	3	K	K
B	D	4	B	3	B	B	B	B	B
C	C	T	C	C	C	C	C	C	C
D	D	2	D	T	D	T	D	E	D
E	E	E	E	E	E	E	E	E	E
F	1	F	S	S	M	F	S	S	S
G	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	7	J	K	J	K	J	K	J
K	K	K	K	K	K	K	K	K	K
L	A	L	L	E	L	L	L	L	L
M	N	N	M	N	N	M	N	M	N
N	M	N	N	N	N	N	N	N	N
O	O	E	O	E	O	A	O	A	O
P	2	E	T	T	D	P	T	T	T
Q	2	Q	2	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	R	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	2	T	G	T	T	T	G	T
U	U	U	U	U	U	U	U	U	U
V	D	E	D	E	D	V	D	O	E
W	W	W	W	W	W	W	W	W	W
X	X	X	H	X	X	X	X	X	X
Y	I	Y	Y	Y	1	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
Wrong	14	12	10	14	7	4	9	10	8

TABLE 26 ESIR RESULTS FOR TOM2 VS TOM13

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	Y	1	Y	I	1	1	1	I	1
2	Q	2	2	T	2	2	2	2	2
3	K	K	3	K	3	K	3	3	3
4	I	R	I	7	4	I	4	1	I
5	5	O	5	9	5	5	5	Y	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	A	8	8	8	8	8	8	8	8
9	I	9	9	9	9	9	9	9	9
A	A	E	A	A	A	E	A	A	A
B	E	V	V	V	D	D	T	P	V
C	C	C	C	C	C	C	C	C	C
D	D	O	D	D	D	D	D	D	D
E	E	V	P	V	V	V	P	V	E
F	F	F	F	F	F	S	F	F	F
G	G	G	G	T	G	G	G	G	G
H	X	H	H	H	H	H	H	H	H
I	I	I	I	I	I	I	I	I	I
J	J	J	J	K	J	K	J	K	J
K	K	K	K	3	K	K	K	K	K
L	L	P	L	V	L	L	L	L	L
M	2	N	N	M	M	M	M	M	M
N	M	M	M	M	N	N	M	M	M
O	O	O	O	V	O	B	O	V	O
P	T	K	T	2	D	D	P	D	P
Q	Q	C	J	2	Q	C	Q	C	Q
R	R	P	4	I	R	Y	R	R	R
S	S	S	S	S	S	S	S	S	S
T	T	K	P	K	T	D	T	D	T
U	Q	U	U	U	U	U	U	U	U
V	E	E	V	V	V	V	V	V	V
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	I	Y	I	Y	Y	I	Y	Y	Y
Z	Z	Z	Z	Z	Z	6	Z	Z	Z
Wrong	14	15	11	16	3	14	4	9	2

TABLE 27 ESIR RESULTS FOR TOM1 VS TOM23

	LPPS	DPPS	LFCC	DFCC	MFMC	DFMC	BLMC	DLMC	Fusion
0	0	0	0	0	0	0	0	0	0
1	H	1	M	1	Y	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	A	3	3	3	3	K	3	3	3
4	R	4	R	4	R	4	R	4	4
5	5	Y	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	H	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
A	H	A	A	A	A	A	A	A	A
B	D	D	D	D	D	E	D	D	D
C	C	C	C	C	C	C	C	C	C
D	P	P	T	D	P	D	D	P	D
E	E	E	E	E	E	T	V	E	E
F	F	F	F	F	F	F	F	F	F
G	G	0	C	C	C	G	G	0	C
H	8	H	X	H	H	H	H	H	H
I	9	9	I	9	I	I	I	I	I
J	J	0	2	G	J	J	K	K	J
K	K	2	K	P	K	K	K	K	K
L	L	L	L	A	L	L	L	L	L
M	N	N	N	N	N	N	N	N	N
N	I	N	N	N	N	N	N	N	N
O	O	V	O	E	O	E	O	E	O
P	P	P	T	B	P	D	P	P	P
Q	Q	2	Q	Q	Q	2	Q	2	Q
R	5	R	R	V	R	R	R	4	R
S	S	G	S	S	S	F	S	S	S
T	T	P	T	P	T	T	T	T	T
U	U	U	U	U	U	U	U	U	U
V	2	E	V	E	V	E	V	E	E
W	W	W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	1	Z	Z	Z	Z	Z	Z	Z
Wrong	13	14	9	12	6	8	5	9	4

94.4%. The fusion results usually did not do better than the best feature's results, with tables 3, 26, and 27 being the exceptions. The fusion result was always much closer to the best feature than the worst feature. Table 26 shows a classic case where feature fusion results were significant. Five of the features performed miserably, two features performed well and the fusion results only missed two words. The regular features out performed their delta feature counterparts. The best feature was usually the MFMC, followed by BLMC and LFCC. The best Delta feature was DFMC. Using templates increased the average recognition accuracy of the recognizers.

5.2.2 Optimal Recognition Tests

Tables 28 through 36 show the results for the optimal combination of recognizers using the best features. This recognition test is a two step process. The first step uses the DTW recognizer and the features: LPPS, LFCC, MFMC, DFMC, and BLMC. If all the recognition results in the first step agree as to the correct answer, then the recognition result stands. If they don't agree, then a second set of recogntiion tests are performed on a newly created sublist. The sublist consists of the top 6 words with the smallest error distances from the fusion result in step 1. The additional recognition tests performed in step 2 are the FIR using the features: LFCC, MFMC, and BLMC; and the ESIR using the feature MFMC. After the additional recognition tests are run all of the results from step 1 and 2 are fused together. This result is presented as the recognition result.

The tables for the optimal combination test are presented similarly to the tables for the previous test results with one difference. The columns for the step 2 recognizers are blacked out when the recognizers in the first step all agreed and thus the second step was not performed.

The recognition results were generally very good. All but one test was better than 91.6% accurate. The one outlier was 86.1% accurate. The results generally support the hierarchical approach to speech recognition. Examples where the results from the second step

TABLE 28 HIERARCHICAL RESULTS FOR TOM2 VS TOM1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusio
0	0	0	0	0	0					0
1	Y	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	V	T	P	D	P	E	P	P	P	P
C	C	C	C	C	C					C
D	D	D	D	B	D	D	D	D	D	D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	M	M	N	N	M	M	N	M	N	N
O	O	O	O	O	O					O
P	T	T	P	D	P	T	P	P	D	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	Q	U	U	U	U	U	U	U	U	U
V	E	V	V	V	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	6	3	1	3	2					1

TABLE 29 HIERARCHICAL RESULTS FOR TOM3 VS TOM1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	7	5	5	5	I	Y	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	D	B	B	D	B	B	B	B	B	B
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	T	G	G	G	T	G	G	C	G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	K	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	T	T	T	T	P	T	P	T
Q	W	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	D	D	D	V	D	D	D	D	B	D
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	3	2	2	3	2					2

TABLE 30 HIERARCHICAL RESULTS FOR TOM1 VS TOM2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	M	1	1	1	1	1	1	1	1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	D	E	D	E	E	D	E	D	D	D
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	N	M	M	N	N	M	M	N	M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	B	P	P	P	P	B	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	G	T	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	E	V	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	2	3	3	2	2					1

TABLE 31 HIERARCHICAL RESULTS FOR TOM3 VS TOM2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	I	5	5	5	5	5	5	I	I	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	T	D	D	K	D	D	E	D	D	D
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	E	E	E					E
F	K	L	F	F	L	S	F	S	M	L
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	N	M	M	N	M	M	M	N	N
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	D	D	D	E	D	D	E	D	E	D
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	G	T	T	S	T	T	T	T	T	T
U	U	U	U	U	U					U
V	D	D	D	V	D	D	D	D	D	D
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	I	Y	Y	Y	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z					Z
WRONG	8	5	3	3	5					5

TABLE 32 HIERARCHICAL RESULTS FOR TOM1 VS TOM3

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	Y	5	5	5	5	Y	I	Y	I	Y
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	B	B	P	E	B	B	V	B	V	V
C	C	C	C	C	C					C
D	D	D	V	T	V	D	P	D	D	D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	K	K	J	J	K	K	J	J	7	J
K	K	K	K	K	K					K
L	L	L	L	F	L	L	L	L	L	L
M	M	M	M	M	M					M
N	N	N	N	M	N	M	M	N	N	N
O	O	O	O	O	O					O
P	P	P	P	T	P	P	P	P	P	P
Q	2	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	P	P	P	E	P	P	P	P	P	P
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	4	2	3	6	3					3

TABLE 33 HIERARCHICAL RESULTS FOR TOM2 VS TOM3

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	V	V	V	T	V	V	D	T	P	V
C	C	C	C	C	C					C
D	V	V	V	D	D	V	V	D	P	V
E	E	E	E	E	E					E
F	X	F	F	S	F	F	F	F	F	F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	Y	I	I	Y	I	I	I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	N	M	M	M	M	M	M	M	M
N	N	N	N	M	N	N	N	N	N	N
O	O	O	O	O	O					O
P	V	V	P	T	T	T	P	P	P	T
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	E	V	V	E	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	6	4	2	6	2					3

TABLE 34 HIERARCHICAL RESULTS FOR TOM3 VS TOM12

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	I	5	5	7	5	5	5	I	I	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	D	B	B	D	B	B	B	B	B	B
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	E	E	E					E
F	S	F	F	F	F	F	F	F	M	F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	K	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	M	M	M	M	M	M	M	N	M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	D	T	D	T	T	T	D	T	D	T
Q	W	2	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	D	D	D	V	D	D	D	D	D	D
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	7	3	2	3	2					2

TABLE 35 HIERARCHICAL RESULTS FOR TOM2 VS TOM13

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	Y	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2					2
3	3	3	3	K	3	3	3	3	3	3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	V	V	D	D	P	V	D	P	D	P
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	M	N	N	N	N	N	N	N	N	N
O	O	O	O	B	O	O	O	O	O	O
P	P	P	P	D	P	T	P	P	D	P
Q	Q	Q	Q	2	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	D	T	T	T	T	T	T
U	Q	U	U	U	U	U	U	U	U	U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	4	1	1	6	1					1

TABLE 36 HIERARCHICAL RESULTS FOR TOM1 VS TOM23

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	D	E	E	E	E	D	E	E	D	D
C	C	C	C	C	C					C
D	D	D	D	P	D	D	P	D	P	D
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	K	J	J	K	J	J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	N	M	M	N	N	M	M	N	N
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	D	P	P	P	P	P	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	G	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	E	E	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	2	2	2	5	3					2

maue the difference in the fusion results are words N and P in Table 28, word P in Table 29, word M in Table 30, word M in Table 31, words D and J in Table 32, and word P in Table 33. The only negative example is the word B in Table 32.

5.3 Talker Independent

The talker independent tests are designed to test a talker's words against a template of multiple talkers. Two sets of tests are run. The first set is to test the creation of the templates and the second to test the updating of the templates.

5.3.1 Creating Templates

Two sets of templates were created, each consisting of four talkers. Template1 consists of the words from tom1, bud1, bill1, and steve1. Template2 consists of the words from tom2, bud2, bill2 and steve 2. The talkers words are first calibrated to baseline their performance. The two step recognition approach used in Section 5.2.2 was used for the calibration test. Tables 37 through 42 show the results for bud, bill, and steve. The results from the previous section are used for the calabration test on tom.

Tables 43 through 50 show the results for the talker independent tests using the optimal recognition technique. The results for the talker independent tests are varied. The tests for tom1 and tom2 produced recognition accuracies of 94.4% and 97.2% correct, respectively. These results may be an artifact of the template creating process. The tests for bud1 and bud2 produced recognition accuracies of 83.3% and 80.5% respectively. The tests for bill1 and bill2 produced recognition accuracies of 83.3% for both. The tests for steve1 and steve2 produced recognition accuracies of 86.1% and 83.3% respectively. The consistant results of the last three talkers suggest that these accuracies are typical for talker independent recognition results.

5.3.2 Updating templates

Templates are updated as a function of the update_factor. Two update_factors will be tested, one for updating the template quickly and one for updating the template slowly.

TABLE 37 CALIBRATION RESULTS FOR BUD2 VS BUD1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	P	A	P	A	P	D	A	A	P	A
C	C	C	C	C	C					C
D	P	T	T	D	T	T	P	T	P	T
E	E	D	P	P	E	E	P	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	J	G	G	G	G	G	G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	N	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	P	P					P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	2	3	3	4	3					2

TABLE 38 CALIBRATION RESULTS FOR BUD1 VS BUD2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	K	A	A	A	A	A	A	A	A	A
B	V	V	P	T	V	V	P	V	V	V
C	C	C	C	C	C					C
D	V	V	P	V	V	V	P	V	V	V
E	P	E	E	E	E	E	E	E	E	E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	N	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	P	P					P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	P	T	T	T	D	T	T	T	T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	S	S	X	X	X	S	X	X	S	S
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	5	4	2	3	2					3

TABLE 39 CALIBRATION RESULTS FOR B1LL2 VS BILL1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	I	5	5	Y	5	5	5	5	5	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	H	A	A	A	A	A	A	A	A	A
B	B	B	B	V	B	V	B	B	T	B
C	C	C	C	C	C					C
D	B	A	T	T	T	A	T	A	T	T
E	T	P	G	E	V	T	E	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	T	G	T	G	V	G	Z	V	Z	V
H	H	H	H	H	H					H
I	Y	Y	I	I	I					I
J	J	J	J	J	J					J
K	K	J	K	J	K	K	K	K	K	K
L	L	L	L	L	L					L
M	M	N	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	T	T	T	T	T	T	T	T
Q	Q	2	Q	Q	Q	2	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	V	T	T	T	T	T	T
U	U	U	U	U	U					U
V	B	V	V	V	V	V	V	B	B	V
W	W	W	W	W	W					W
X	X	X	X	X	F	X	X	F	S	X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	7	5	4	7	5					3

TABLE 40 CALIBRATION RESULTS FOR B1LL1 VS BILL2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	Y	Y	5	Y	Y	5	5	5	Y	Y
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	V	V	B	B	V	V	D	V	D	V
C	C	C	C	C	C					C
D	B	B	B	G	B	B	B	B	P	B
E	E	E	E	E	E					E
F	F	F	F	F	F					F
G	E	G	G	G	G	G	G	G	G	G
H	H	H	H	H	H					H
I	Y	Y	I	I	I					I
J	J	K	J	J	J	K	J	J	J	J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	N	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	V	T	V	P	P	T	P	P
Q	U	2	2	2	2	2	2	2	2	2
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	P	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	V	C	V	V	V	V	G	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	5	5	3	6	5					4

TABLE 41 CALIBRATION RESULTS FOR STEVE2 VS STEVE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	A	8	8	H	8	8	8
9	9	9	Y	J	9	9	I	9	J	9
A	A	A	A	G	A	A	A	A	A	A
B	B	B	B	B	B					B
C	C	C	C	C	C					C
D	D	D	D	P	D	D	D	D	D	D
E	E	E	E	D	E	E	E	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	T	G	G	G	G	G	G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	K	K	J	K	K	K	K	J	K
L	L	L	L	L	L					L
M	M	N	M	N	M	M	M	M	M	M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	P	P					P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	G	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	V	P	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	0	0	1	11	0					0

TABLE 42 CALIBRATION RESULTS FOR STEVE1 VS STEVE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	0	7	5	4	0	4	0	7
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	H	H	H	H	H	H	H	H
9	9	9	9	9	9					9
A	A	A	H	A	H	A	H	H	H	A
B	B	B	B	D	B	B	P	D	P	B
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	E	E	B	P	E	E	B	E	E	E
F	F	F	F	F	F					F
G	G	G	G	T	G	G	G	G	G	G
H	H	H	H	H	H					H
I	I	I	Y	I	I	I	I	I	I	I
J	K	K	J	J	K					J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	N	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	P	P					P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	G	T	T	T	T	G	T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	C	Z	Z	Z	Z	Z	Z
WRONG	1	1	5	7	3					2

TABLE 43 TEMPLATE RESULTS FOR TOM2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	Y	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	3	8	8	H	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	V	V	P	V	V	P	V	P	P	P
C	C	C	C	C	C					C
D	D	D	D	D	D					D
E	V	V	D	D	D	V	D	D	D	D
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	Y	Y	I	I	5	Y	I	I	I	I
J	J	J	J	K	J	J	J	J	J	J
K	J	K	K	K	K	K	K	K	K	K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	T	P	P	P	T	P	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	V	V	V	D	V	V	V	V	P	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	6	3	2	8	3					2

TABLE 44 TEMPLATE RESULTS FOR TOM1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	R	4	4	4	R	4	4	R	4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	3	8	8	H	8					8
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	D	D	D	D	D					P
C	C	C	C	C	C					C
D	D	D	D	V	P	D	D	D	D	D
E	E	T	D	E	E	E	D	B	E	E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	K	J	J	J	K	J	J	J	J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	N	N	N	M	N	N	M	N	N	N
O	O	O	O	O	O					O
P	P	P	P	P	P					P
Q	Q	Q	Q	Q	Q					Q
R	I	R	R	R	4	R	R	R	R	R
S	S	S	S	S	S					S
T	T	T	P	T	P	T	P	P	T	T
U	U	U	U	U	U					U
V	B	V	B	V	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	4	3	4	3	5					1

TABLE 45 TEMPLATE RESULTS FOR BUD2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0	■	■	■	■	0
1	Y	Y	1	1	Y	1	1	1	1	1
2	Q	Q	Q	Q	Q	■	■	■	■	Q
3	J	J	3	J	J	K	3	3	3	3
4	4	4	4	4	4	■	■	■	■	4
5	I	5	5	7	5	M	5	5	Y	5
6	6	6	6	6	6	■	■	■	■	6
7	7	7	7	7	7	■	■	■	■	7
8	8	8	8	H	8	8	8	8	H	8
9	9	9	9	9	9	■	■	■	■	9
A	K	A	H	K	A	A	A	A	A	A
B	K	A	V	D	A	H	A	A	V	A
C	C	C	C	C	C	■	■	■	■	C
D	J	D	D	T	D	D	D	D	T	D
E	V	V	D	P	D	V	D	P	D	D
F	F	F	S	F	F	F	F	F	F	F
G	C	G	G	C	G	G	G	G	G	G
H	H	H	H	H	H	■	■	■	■	H
I	I	I	I	I	I	■	■	■	■	I
J	J	J	K	K	K	J	K	K	7	J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	N	N	M	N	N	N	N	M	N	N
N	N	N	N	N	N	■	■	■	■	N
O	L	L	O	O	O	L	O	O	4	O
P	D	D	D	T	D	D	D	D	V	D
Q	Q	Q	Q	Q	Q	■	■	■	■	Q
R	R	R	R	R	R	■	■	■	■	R
S	S	S	S	S	S	■	■	■	■	S
T	K	T	T	T	K	T	T	T	T	T
U	0	Q	U	Q	Q	U	U	U	U	U
V	C	G	T	C	J	G	P	T	T	G
W	W	W	W	W	W	■	■	■	■	W
X	6	X	X	6	X	X	X	X	X	X
Y	I	I	Y	Y	5	Y	Y	Y	Y	Y
Z	C	Z	Z	C	Z	Z	Z	Z	Z	Z
WRONG	18	11	8	16	12					7

TABLE 46 TEMPLATE RESULTS FOR BUD1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0	■	■	■	■	0
1	1	1	1	1	1	■	■	■	■	1
2	Q	2	2	2	2	2	2	2	2	2
3	0	0	3	J	7	0	3	7	0	7
4	4	4	4	G	4	4	4	4	2	4
5	5	5	5	5	5	■	■	■	■	5
6	6	6	6	6	6	■	■	■	■	6
7	7	7	7	7	7	■	■	■	■	7
8	8	8	8	8	8	■	■	■	■	8
9	9	9	I	7	I	9	5	I	3	9
A	J	A	A	K	A	A	8	A	A	A
B	V	V	V	3	V	V	B	V	V	V
C	C	C	C	C	C	■	■	■	■	C
D	V	V	B	B	V	V	B	V	V	V
E	E	P	E	E	E	P	E	E	P	E
F	S	S	F	F	F	S	X	F	X	S
G	J	G	B	7	E	B	B	V	Q	G
H	H	H	H	H	H	■	■	■	■	H
I	9	9	I	I	I	9	I	I	I	I
J	J	J	J	J	J	■	■	■	■	J
K	J	J	J	J	J	■	■	■	■	J
L	9	L	L	L	L	L	L	L	L	L
M	N	N	M	M	M	N	M	M	M	M
N	N	N	N	N	N	■	■	■	■	N
O	O	O	O	O	O	■	■	■	■	O
P	A	P	D	B	A	P	P	A	V	P
Q	Q	Q	Q	Q	Q	■	■	■	■	Q
R	1	R	R	4	R	R	R	R	5	R
S	S	S	S	S	S	■	■	■	■	S
T	G	G	T	G	J	G	T	T	G	G
U	U	U	U	U	U	■	■	■	■	U
V	V	V	V	C	V	V	B	V	V	V
W	W	W	W	W	W	■	■	■	■	W
X	X	X	X	S	X	X	X	X	X	X
Y	Y	9	Y	Y	Y	I	Y	Y	Y	Y
Z	C	C	Z	Z	C	Z	Z	Z	Z	Z
WRONG	15	11	7	13	9					6

TABLE 47 TEMPLATE RESULTS FOR B1LL2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	7	7	0	0	7	0	0	7	7	0
1	1	Y	1	1	1	1	1	1	1	1
2	Q	2	2	Q	2	2	Q	2	Q	2
3	C	V	3	3	3	V	3	3	3	3
4	4	4	4	4	4					4
5	I	I	I	I	I					I
6	6	6	6	6	6					6
7	7	J	7	7	7	J	7	7	7	7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	H	V	H	A	A	V	H	A	H	A
B	V	V	B	B	V	V	B	D	B	V
C	C	C	C	C	C					C
D	D	B	D	D	B	B	D	B	D	B
E	D	B	E	D	B	E	E	E	E	E
F	F	F	F	F	F					F
G	T	G	T	G	T	P	T	T	T	G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	J	K	K	K	T	K	K	K	K
L	L	L	L	L	L					L
M	J	J	M	M	M	M	M	M	M	M
N	N	J	N	M	N	N	N	N	N	N
O	L	L	O	O	L	O	O	O	O	O
P	P	V	D	D	P	P	D	P	D	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	F	S	S	S	S	S	S
T	T	V	D	T	P	P	D	T	D	T
U	Q	Q	U	Q	Q	Q	U	U	U	Q
V	V	V	B	V	V	D	D	D	V	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	I	I	Y	I	I	I	Y	Y	Y	I
Z	V	V	P	Z	P	P	P	P	P	P
WRONG	13	19	7	9	12					6

TABLE 48 TEMPLATE RESULTS FOR B1LL1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	Y	1	1	1	1	1	1	1	Y	1
2	2	2	2	2	2					2
3	A	V	3	3	A	A	3	A	3	A
4	4	4	4	4	4					4
5	I	5	I	5	I	5	I	I	I	I
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	V	8	E	V	P	8	A	8	A
B	V	V	E	V	E	V	E	E	P	V
C	C	C	C	C	C					C
D	V	V	V	V	V					V
E	E	E	E	D	E	P	E	E	E	E
F	F	F	F	S	F	W	F	F	F	F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	J	J	J	J	J					K
L	9	L	L	L	L	L	L	L	L	L
M	N	M	M	K	W	M	M	W	N	M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	P	P	P	E	P	P	E	P	P	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	G	P	P	E	P	P	P	P	P	P
U	U	U	Q	U	Q	U	U	U	U	U
V	V	V	P	V	P	P	P	T	P	V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	9	5	8	9	11					6

TABLE 49 TEMPLATE RESULTS FOR STEVE2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	7	0	0	0	0	0	7	0
1	1	1	1	1	1					1
2	2	2	2	2	2					2
3	3	3	3	3	3					3
4	4	0	4	4	4	0	4	4	7	0
5	7	7	7	J	7	7	7	7	7	7
6	6	6	6	6	C	6	6	Z	7	6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	E	E	T	E	B	B	B	A	B
B	E	B	E	B	E	B	E	E	B	B
C	C	C	C	C	C					C
D	E	D	E	B	E	D	E	E	B	E
E	E	E	E	B	E	E	E	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	T	G	G	G	G	G	G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	J	J	J	J	J					J
K	K	J	K	J	J	J	K	K	J	J
L	L	L	L	L	L					L
M	N	N	M	M	M	N	M	M	M	M
N	9	N	N	N	N	N	N	N	N	N
O	O	O	O	O	O					O
P	B	B	E	B	E	B	E	E	B	B
Q	G	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	F	S	S	S	S	X	S
T	T	G	G	T	G	T	G	G	G	G
U	V	U	U	U	U	U	U	U	Q	U
V	V	V	E	E	V	V	D	V	D	V
W	W	W	W	B	W	W	W	W	W	W
X	X	X	X	X	S	H	X	S	X	X
Y	Y	Y	Y	I	Y	Y	Y	Y	Y	Y
Z	Z	Z	Z	Z	Z					Z
WRONG	9	7	8	11	9					6

TABLE 50 TEMPLATE RESULTS FOR STEVE1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	Q	2	2	2	Q	Q	Q	2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	7	7	0	C	7	7	0	7	Q	0
6	6	6	6	C	6					6
7	7	7	7	7	7					7
8	8	8	8	8	8					8
9	9	9	9	9	9					9
A	A	E	8	E	A	E	D	D	8	A
B	E	E	E	E	E					E
C	C	C	C	C	C					C
D	E	E	E	E	E					E
E	E	E	E	D	E	E	E	E	E	E
F	F	F	F	F	F					F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	G	J	J	J	J	J	J	J	J	J
K	A	A	J	J	A	J	J	A	J	J
L	L	L	L	L	L					L
M	N	M	M	M	M	M	M	M	M	M
N	N	N	N	N	M	N	N	N	N	N
O	O	O	O	O	O					O
P	P	P	E	T	E	P	E	E	T	P
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	4	R	R	R	R	R	R
S	S	S	S	S	X	S	S	X	S	S
T	G	G	G	G	G					G
U	Q	Q	Q	Q	U	U	U	U	U	U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	C	C	Z	C	C	C	C	C	C
WRONG	8	8	10	12	9					5

Tables 51 and 52 show the calibration test for the update talker shane. First, the update talker is tested against the templates used in the creating template tests. These results are shown in tables 53 and 54. Then, shane1 is added to template1 and shane2 is added to template2, both have an update_factor of 4.0. Using this update_factor averages all five talkers equally. Tables 55 and 56 show the results for the slow tracking template update tests. Finaly, the templates from the creating template tests are recreated and updated with shane1 and shane2 using an update_factor of 1.0. Using this update_factor averages the template words with the update words. Tables 57 and 58 show the results from the fast tracking update templates test.

The not in template results are very significant. This talker achevied accuracies of 83.3% and 86.1% correctness. This is similar to the talker independent results with the talkers in the template. This indicates that a respresentive template may be able to achieve talker independent recognition accuracies in the 80% range for a large number of talkers. The slow tracking results achieved the same accuracy as the not in template test. This is expected since this test is very similar to the previous talker independent tests. The fast tracking results produced accuracies of 83.3% and 88.8%. These results at first appear strange when compared to the slow tracking results. The shane2 vs template1 test showed decreased preformance. However, this can be explained by looking at the calabration test for shane2 vs shane1, which had an accuracy of 83.3%. The fast tracking updating moves the template very close to the new talker's words. The shane1 vs template2 test also supports this idea.

5.4 Conclusions

Considering the difficulty of the word set, the recognition accuracies are respectable. The DTW appears to be the best of the recognizers followed by the FIR. The MFMC features usually out performed the other features for the talker dependent tests, but not for the talker independent tests. The two step recognition process usually helps the recognition

TABLE 51 CALIBRATION RESULTS FOR SHANE2 VS SHANE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	Q	Q	Q	2	Q	Q	2	2	2	Q
3	3	3	3	3	3					3
4	R	R	4	K	R	4	4	4	1	4
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	B	H	H	H	H	H	H	H	H
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	0	0	H	V	0	G	H	E	H	H
C	C	C	C	C	C					C
D	E	D	D	D	D	D	D	D	D	D
E	E	E	E	E	E					E
F	S	F	F	F	F	F	F	F	F	F
G	C	G	G	C	G	G	T	Z	Z	G
H	H	H	H	H	H					H
I	I	I	Y	I	I	I	Y	I	9	I
J	J	J	J	J	J					J
K	J	J	J	J	K	K	J	K	J	J
L	L	L	L	L	L					L
M	N	N	M	M	N	M	M	M	M	M
N	N	N	N	N	N					N
O	O	O	O	H	O	O	O	O	O	O
P	E	P	D	D	B	P	D	B	D	D
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	C	C	V	V	V	Z	V	V	V	V
WRONG	10	7	7	8	6					6

TABLE 52 CALIBRATION RESULTS FOR SHANE1 VS SHANE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0	■	■	■	■	0
1	1	1	1	1	1	■	■	■	■	1
2	2	2	2	2	2	■	■	■	■	2
3	3	3	3	3	3	■	■	■	■	3
4	4	R	4	R	4	R	R	R	R	R
5	5	5	5	5	5	■	■	■	■	5
6	6	6	6	6	6	■	■	■	■	6
7	7	7	7	7	7	■	■	■	■	7
8	8	8	H	8	A	A	H	H	H	8
9	9	9	9	9	9	■	■	■	■	9
A	A	A	H	8	A	A	A	A	A	A
B	D	8	P	D	P	8	E	P	E	P
C	C	C	C	C	C	■	■	■	■	C
D	D	D	P	D	P	P	D	D	8	D
E	E	E	E	E	E	■	■	■	■	E
F	F	F	F	F	F	■	■	■	■	F
G	G	G	G	G	G	■	■	■	■	G
H	H	H	H	H	H	■	■	■	■	H
I	I	I	I	I	I	■	■	■	■	I
J	J	J	J	J	J	■	■	■	■	J
K	K	K	K	K	J	K	K	K	K	K
L	L	L	L	L	L	■	■	■	■	L
M	M	M	M	M	F	M	M	M	M	M
N	N	N	N	N	N	■	■	■	■	N
O	O	O	O	O	O	■	■	■	■	O
P	P	P	P	K	P	P	P	P	D	P
Q	Q	Q	Q	Q	Q	■	■	■	■	Q
R	R	R	R	R	R	■	■	■	■	R
S	S	S	S	S	S	■	■	■	■	S
T	P	T	G	G	G	T	G	G	G	G
U	U	U	U	U	U	■	■	■	■	U
V	V	V	V	V	V	■	■	■	■	V
W	W	W	W	W	W	■	■	■	■	W
X	X	X	X	X	X	■	■	■	■	X
Y	Y	Y	Y	Y	Y	■	■	■	■	Y
Z	Z	Z	Z	Z	Z	■	■	■	■	Z
WRONG	2	2	5	5	6					3

TABLE 53 NOT IN TEMPLATE RESULTS FOR SHANE2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	Y	Y	1	1	1	Y	1	1	1	1
2	2	2	2	Q	Q	2	2	2	2	2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	7	7	7	7	7	7	7	7	7
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	D	8	H	H	H	H	8
9	9	I	9	9	9	I	9	9	9	9
A	A	A	A	A	A					A
B	2	2	G	V	Z	Z	C	Z	X	Z
C	C	C	C	C	C					C
D	B	B	D	D	B	B	D	B	D	B
E	B	E	B	D	B	E	E	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	T	T	T	G	T	T	J	T
H	H	H	H	H	H					H
I	Y	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J					J
K	K	K	K	J	K	K	K	K	K	K
L	L	M	L	L	L	L	L	L	L	L
M	M	M	M	M	M					M
N	N	N	M	N	N	N	N	N	M	N
O	O	O	O	O	O					O
P	B	B	D	E	B	B	D	P	D	D
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	X	S	S	S	X	S	X	S
T	P	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	V	Z	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	6	X	X	X	X	X	X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	8	7	7	11	7					5

TABLE 54 NOT IN TEMPLATE RESULTS FOR SHANE1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	Q	Q	2	2	Q	2	Q	2
3	3	3	3	3	3					3
4	O	R	4	4	R	R	R	R	R	R
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	8	H	H	8	H	8	8
9	9	9	9	9	9					9
A	H	A	A	A	A	A	A	A	A	A
B	8	P	D	V	P	P	D	P	D	D
C	G	C	C	C	C	C	C	C	C	C
D	D	P	P	E	P	P	D	P	D	D
E	D	P	P	D	D	D	D	D	D	D
F	F	F	F	S	F	F	F	F	X	F
G	G	G	G	J	G	G	G	G	G	G
H	H	H	H	H	H					H
I	5	5	I	I	5	5	I	I	I	I
J	J	J	J	J	J					J
K	2	2	J	J	J	J	J	J	J	J
L	L	L	L	L	L					L
M	F	F	M	M	F	F	M	F	L	M
N	M	M	N	F	M	M	M	M	M	M
O	O	O	O	F	O	R	O	O	O	O
P	P	D	P	G	D	D	D	D	D	D
Q	2	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	P	T	T	T	P	T	T	G	T	T
U	U	U	U	U	U					U
V	V	V	V	Z	V	V	V	V	Z	V
W	W	W	W	W	W					W
X	X	X	F	X	X	X	X	X	X	X
Y	1	Y	Y	1	I	Y	1	Y	Y	Y
Z	Z	2	Z	7	Z	G	Z	Z	T	Z
WRONG	12	10	6	12	12					6

TABLE 55 SLOW TRACKING RESULTS FOR SHANE2 VS TEMPLATE1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	Y	1	1	1	1	1	1	1	1	1
2	Q	Q	2	Q	Q	2	2	2	2	2
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	7	I	7	7	7	5	5	7	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	A	8	H	H	H	H	H
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	Z	Z	C	V	Z	Z	C	Z	X	Z
C	C	C	C	C	C					C
D	B	B	D	D	B	B	D	B	D	B
E	E	E	E	E	E					E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	T	T	Z	G	T	Z	Z	T
H	H	H	H	H	H					H
I	Y	I	I	I	I	I	I	I	I	I
J	J	J	J	J	J					J
K	K	K	K	J	K	K	K	K	K	K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	N	N	N	M	N	N	N	N	M	N
O	O	O	O	O	O					O
P	B	B	D	E	B	B	D	P	D	D
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	X	S	S	S	X	S	X	S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	6	6	X	X	X	X	X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	6	5	5	10	7					5

TABLE 56 SLOW TRACKING RESULTS FOR SHANE1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	2	2	Q	Q	2	2	Q	2	Q	2
3	3	3	3	3	3					3
4	O	R	4	4	Y	R	R	R	R	R
5	5	5	5	5	5					5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	H	8	H	H	H	H	8	8
9	I	I	9	9	9	I	9	9	I	9
A	H	A	A	A	A	A	A	A	A	A
B	8	P	D	D	P	P	D	A	D	D
C	G	C	C	C	C	C	C	C	C	C
D	D	D	P	E	P	D	D	P	D	D
E	D	P	P	D	P	E	D	D	E	D
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	G	G					G
H	H	H	H	H	H					H
I	I	I	I	I	I					I
J	2	2	K	J	K	K	K	J	J	J
K	K	K	K	K	K					K
L	L	L	L	L	L					L
M	F	F	M	N	F	F	M	M	L	M
N	M	M	N	M	N	M	M	M	M	M
O	O	O	O	O	O					O
P	P	D	D	D	D	D	D	D	D	D
Q	2	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	D	T	T	T	T	T	T	T	T	T
U	U	U	U	U	U					U
V	V	V	V	Z	V	V	V	V	V	V
W	W	W	W	W	W					W
X	X	X	X	F	X	X	X	S	X	X
Y	Y	Y	Y	Y	Y					Y
Z	Z	T	Z	G	Z	G	Z	Z	G	G
WRONG	11	9	7	14	7					6

TABLE 57 FAST TRACKING RESULTS FOR SHANE2 VS TEMP1

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0					0
1	1	1	1	1	1					1
2	Q	Q	2	2	Q	2	2	2	2	Q
3	3	3	3	3	3					3
4	4	4	4	4	4					4
5	5	5	5	J	5	5	5	5	5	5
6	6	6	6	6	6					6
7	7	7	7	7	7					7
8	8	8	8	A	8	H	H	H	H	H
9	9	9	9	9	9					9
A	A	A	A	A	A					A
B	Z	X	C	V	Z	Z	C	Z	X	X
C	C	C	C	C	C					C
D	B	B	D	D	B	B	D	B	P	B
E	E	E	E	E	E					E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	T	T	Z	G	T	Z	T	T
H	H	H	H	H	H					H
I	Y	I	I	I	I	I	I	I	I	I
J	J	J	J	K	J	J	J	J	J	J
K	K	K	K	J	K	K	K	K	K	K
L	L	L	L	L	L					L
M	M	M	M	M	M					M
N	N	N	N	N	N					N
O	O	O	O	O	O					O
P	B	B	D	E	B	B	D	B	D	D
Q	Q	Q	Q	Q	Q					Q
R	R	R	R	R	R					R
S	S	S	S	S	S					S
T	T	T	T	T	T					T
U	U	U	U	U	U					U
V	V	V	V	V	V					V
W	W	W	W	W	W					W
X	X	X	X	X	X					X
Y	Y	Y	Y	Y	Y					Y
Z	Z	Z	Z	Z	Z					Z
WRONG	5	4	3	8	5					6

TABLE 58 FAST TRACKING RESULTS FOR SHANE1 VS TEMPLATE2

	LPPS TW	LFCC TW	MFMC TW	DFMC TW	BLMC TW	LFCC 2D	MFMC 2D	BLMC 2D	MFMC ES	Fusion
0	0	0	0	0	0	■	■	■	■	0
1	1	1	1	1	1	■	■	■	■	1
2	2	2	Q	Q	2	2	Q	2	Q	2
3	3	3	3	3	3	■	■	■	■	3
4	O	R	4	4	4	R	R	R	R	R
5	5	5	5	5	5	■	■	■	■	5
6	6	6	6	6	6	■	■	■	■	6
7	7	7	7	7	7	■	■	■	■	7
8	8	8	H	8	H	H	H	H	8	8
9	I	I	9	9	9	I	9	9	9	9
A	A	A	A	A	A	■	■	■	■	A
B	8	P	D	D	P	P	D	A	8	D
C	C	C	C	C	C	■	■	■	■	C
D	D	D	D	D	P	D	D	D	D	D
E	D	E	E	D	E	E	E	E	E	E
F	F	F	F	S	F	F	F	F	F	F
G	G	G	G	G	G	■	■	■	■	G
H	H	H	H	H	H	■	■	■	■	H
I	I	I	I	I	I	■	■	■	■	I
J	K	J	J	J	J	J	J	J	J	J
K	2	K	K	J	K	K	K	K	K	K
L	L	L	L	L	L	■	■	■	■	L
M	M	M	M	M	F	F	M	M	M	M
N	N	N	N	N	N	■	■	■	■	N
O	O	O	O	O	O	■	■	■	■	O
P	P	P	P	G	D	P	D	D	D	D
Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
R	R	R	R	R	R	■	■	■	■	R
S	S	S	S	S	S	■	■	■	■	S
T	T	T	T	G	T	T	T	T	T	T
U	U	U	U	U	U	■	■	■	■	U
V	V	V	V	Z	V	V	V	V	Z	V
W	W	W	W	W	W	■	■	■	■	W
X	X	X	X	F	X	X	X	S	X	X
Y	Y	Y	Y	Y	Y	■	■	■	■	Y
Z	Z	Z	G	G	Z	Z	G	Z	G	G
WRONG	6	3	4	9	5					4

performance.The normalization techniques appear to be successfull based on the ability to combine multiple talkers into a single template. The talker independent recognition accuracies are 80% or better. A template may be tracked to a particular talker through updating the template. The next chapter will provide some conclusions and recommendations for this thesis.

VI Conclusions and Recommendations

The ASRT software was successfully developed and the test results recorded in the previous chapter. It was not possible to run an exhaustive set of tests due to the large number of possible combinations with three recognizers and eight features. However, the tests conducted provide a strong basis for making general recommendations and conclusions for future improvements in ASRT.

6.1 Conclusions

The word extraction process is very important to the recognition process. Noise at the beginning and end of the words can degrade recognition accuracies. Words had to be manually extracted to achieve the best results.

6.1.1 Talker Dependent

ASRT achieves high recognition accuracies usually in 90% to 100% range. Some of recognition errors can be contributed to the word extraction process. The words from the E-set are the hardest to recognize. Analysis of the waveforms show that words that begin with a plosive were difficult to extract properly. Many times this beginning sound was missed when extracted. This led to improper recognitions. The Mel cepstrum features generally achieved the highest recognition accuracies. The delta features did not perform as well as the original features. The recognition accuracies for the fusion results were close to the best feature but usually not as good. Templates help to improve recognition results.

6.1.2 Talker Independent

ASRT achieves good recognition accuracies, usually 80% or better. No feature stood out as always being the best. The accuracies of the fusion results were usually as good as the best feature or better. Templates were successfully created with multiple talkers. Four talkers in a template provides a good base for talker independent speech recognition. Templates can be updated to track towards the current talker.

6.2 Recommendations

The following are provided as recommendations for improvements in the ASRT system:

• The ASRT system needs a better endpoint detector for the extraction process.

• Additional features and recognizers should be tried. The new features should not be based on the power spectrum. A larger filter size on the current FIR and ESIR recognizers may be helpful with the delta-features. The LPPS feature may recognize better by using the log of the power spectrum.

• Image segmentation techniques should be tried on the feature images, with the goal of trying to classify attributes of the word. For example, if fricatives are identified from the feature image, then many of the words in the database can be eliminated immediately because they do not contain any fricatives.

• Try to use neural networks to determine the conditional probability of a recognizer producing a correct result. Then, all recognizers with a probability under a threshold can be eliminated so that they do not degrade the fusion results.

• Try additional fusion techniques. Statistical analysis may show a relationship between words, features, and talkers. This may help to weight each feature differently in the fusion process.

Appendix A ASRT Users Manual

Introduction

The purpose of the appendix is to relate the necessary information for someone to user the ASRT software system. The manual will have three sections: how to setup ASRT, how to use the interactive menus, and how to write scripts for batch processing. ASRT can be run on any of the computers in the VAX cluster but HERCULES is preferable.

How to Setup ASRT

ASRT consists of the following files:

```
SPEECH_CONTROL_PANEL.ADA
EXTRACTION_BODY.ADA
EXTRACTION_PACKAGE.ADA
DATABASE_BODY.ADA
DATABASE_PACKAGE.ADA
FEATURE_BODY.ADA
FEATURE_PACKAGE.ADA
RECOGNITION_BODY.ADA
RECOGNITION_PACKAGE.ADA
TEMPLATE_BODY.ADA
TEMPLATE_PACKAGE.ADA
REC_SUPPORT_BODY.ADA
REC_SUPPORT_PACKAGE.ADA
FUSION_BODY.ADA
FUSION_PACKAGE.ADA
FUSION_REC_PACKAGE.ADA
STRING_OPERATIONS_BODY.ADA
STRING_OPERATIONS.ADA
LINKED_LIST_BODY.ADA
LINKED_LIST.ADA
```

To compile the system, the above files should be compiled in reverse order of appearance. Other compilation sequences may work, but reverse order is recommended. The executable file is generated with the following command:

acs link speech_control_panel.

ASRT may be run from any directory in the user's account and the database may also be stored in any directory in the users account. The database must be pointed to for ASRT to know where it is. This is done in the database subsystem. The full directory path must be used, not a relative one. For example, use the path [TRATHBUN.SPEECH.DATABASE] and not [-.DATABASE]. Also, the database can be on another disk or even another system by adding that information to the path name. For example HERCULES::DJA23:[TRATHBUN.SPEECH.DATABASE].

ASRT stores the database files with the following naming convention PREFIX_WORDNAME.DIG. The underscore is part of the prefix. When the user uses the feature subsystem to create features, new files are created with same name, however the DIG ending of the database word is replaced by the acronym of the feature. The use of prefixes lets the user select and group of words in the database as the database library and any other group as the test words. The names of the words in database library are stored in a file named DATABASE.DAT.

How to Use Interactive Menus

All subsystems in ASRT are accessed from the main menu, which is shown on the following page. When the user selects the number of the appropriate subsystem, the new menu for that subsystem appears. When the user quits from a subsystem, the main menu reappears. Quitting a subsystem does not necessarily mean that data generated in that subsystem is cleared. The user should be able to ascertain whether data is cleared or not by noting menu items to specifically clear an item or in some cases the use can view data items to see there values.

```
1 - GOTO EXTRACTION OPERATIONS
2 - GOTO DATABASE OPERATIONS
3 - GOTO FEATURE OPERATIONS
4 - GOTO RECOGNITION OPERATIONS
5 - GOTO FUSION OPERATIONS
6 - GOTO TEMPLATE OPERATIONS
7 - GOTO BATCH OPERATIONS
8 - TO QUIT

CHOICE ->
```

The batch operations is not a separate subsystem. The script interpreter is built into the main procedure. When selecting this item, the user is prompt for the file name of the script to run. Since all script files end with .SCR, this part of the file name does not need to be entered.

Extraction Operations

The extraction subsystem is used to transfer the digitized speech files to the user's database and extract the words in separate files. Speech files are normally recorded and saved on the VAXStation II/GPX named CALVIN. Therefore the default transfer from location is CALVIN::DUA0:[TOMR]. But this may be changed to any computer on the network running DECNET. The source directory should be configured for world read access, but if the user prefers, the user name and password may be encoded into the source name. A path name of that variety would look like CALVIN"user name password"::DUA0:[TOMR]. The destination path is defaulted to [TRATHBUN.SPEECH.DATABASE], but the user may change this in the database subsystem.

The subsystem menu is show on below:

```
 1 - INPUT FILE NAME FOR EXTRACTION
 2 - SET PREFIX FOR EXTRACTED WORDS
 3 - SET INPUT CHANNEL LOCAL OR NETWORK (NETWORK)
 4 - REVIEW CURRENT SETTINGS FOR EXTRACTION
 5 - CREATE LIST OF SPEECH FILE
 6 - TRANSLATE FILE INTO STANDARD FORMAT
 7 - CREATE EXTRACTION FEATURE LIST
 8 - DISPLAY FEATURE LIST
 9 - EXTRACT WORDS OUT OF FILE
10 - MANUALLY EXTRACT WORDS OUT OF FILE
11 - RUN FULL EXTRACTION PROCESS
12 - QUIT EXTRACTION OPERATIONS
 CHOICE ->
```

Item 1 is used for identifying which file is to be used for items 5 and 6.

Item 2 is used for inputting a prefix which will be appended to the front of the file names for the extracted words when using items 9 and 10.

Item 3 is a switch to tell the translation program in item 6 whether to look to the network path or the local database when translating a file. This item is included because digitized speech files can be copy manually to the database from the remote node or the files may have been digitized else where and load directly into the database.

Item 4 is used to display and change certain setting used for automatic extraction of words. Item 4 will be discussed last.

Item 6 needs to be discussed before item 5. Item 6 translates a speech file in the DSC 200 format and creates an ASCII version in the database directory. Also, the translation program makes a linked list of the speech file in memory.

Item 5 also makes a linked list of the speech file, but from the already translated speech file in the database directory. This item make it so that item 6 needs to be done only once for each file. The next time the user needs the linked list of the speech file created, item 5 should be used since it is faster. Plus the file on the remote node may have already been deleted.

Item 7 creates a power spectrum linked list of time slices of the speech linked list. This step needs to be done for the extracted in item 9 and 10 can occur.

Item 8 display the power spectrum linked list created in item 7.

Item 9 automatically extracts and creates words files from the speech list and the power spectrum list. Item 9 is governed by the setting in item 4.

Item 10 is a manually extraction scheme, where the user visually looks at the power spectrum list and decides where the endpoints of the words are. Item 10 has it own sub menu which is show below:

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
000000000022211111211111212211111101111100121234556687789971111146776666666666655544
3121111001011010111000110111211111111166996211146999986533434333322211111111010
111!111111111111111111111233445699999864331101111110666542100011111111111111110

  1 - TO DISPLAY THE WHOLE FILE
  2 - TO DELETE A SECTION
  3 - TO WRITE A SECTION TO A FILE
  4 - TO WRITE A SECTION TO A FILE WITHOUT TRIMMING
  5 - TO UNDO LAST WRITE
  6 - TO CHANGE COEFFICIENT VALUE
  7 - TO CHANGE THE FILTER SIZE
  8 - TO GOTO BEGINNING OF LIST
  9 - TO EXIT
  CHOICE ->
```

The first two lines are included to show the column numbers. The next three lines are the power spectrum list. Item 1 can be used to show the entire list, because this menu only display three lines at a time.

Item 2 is used to deleted as many time slices as the user needs to position the first time slice on the next word to extract as the fist time slice in the characterization list.

After deleting every time slice in front of the word to be extracted the user uses item 3 to extract the word with triming. Item 3 will prompt for the word name and how many time slices are in the word and will then display the numbr of datapoints deleted from the front and back of the word. Remember that a prefix will be attached to the front of the word name and a .DIG ending to the rear, to complete the file name.

The user alternates between items 2 and 3 until all the words are extracted, then uses item 4 to exit back the subsystem menu.

Item 4 is an alternative to item 3. Item 3 will extract the word without triming, but otherwise operates the same.

ITem 5 will moves the pointers in the characterization list and the speech list back to their position before the last use of item 3 or 4. This allows to the user to reextract a word after a mistake or adjustment has been made.

Item 6 allows the user to change the coeeficient value to adjust hte triming process. the coefficient value is the number of standard deviation used in the treshold value.

Item 7 allows the user to change the filter size in the moving average filter to adjust hte triming process.

Item 8 moves the pointers for the characterization list and the speech list back to the beginning to start over.

Item 9 quits the manual extraction menu, but the lists are maintained.

Item 11 of the subsystem menu is just a single item to execute items 6,7, and 9.

Item 12 returns back to the main menu.

When item 4 is selected a new menu is displayed as shown below:

```
1 - THE BEGINNING WORD COUNT IS     6
2 - THE END WORD COUNT IS    12
3 - THE SPEECH WINDOW SIZE IS  256
4 - NETWORK LOCATION IS CALVIN::DUA0:[TOMR]
5 - WORDS PREFIX IS TOM1_
6 - THE EXTRACTION MODE IS NORMAL
7 - TO QUIT REVIEW

NUMBER OF SETTING TO CHANGE ->
```

Items 1 and 2 are used in the heuristic extraction rules for the automatic extraction procedure. After viewing the power spectrum list the user may want to change these two if using the automatic extraction program.

Item 3 is used to change the number of data points in a time slice. This item effects the creation of the power spectrum list.

Item 4 is used to change the remote node path. This item effects the translation program.

Item 5 is used to change the prefix value for extraction. This item effects both extraction procedures.

Item 6 is used to change how the automated extraction program generates the file names for the words to be extracted. Normal, as show, means the user types in the word names. Auto_database means the word names are generated from the databse list. Auto_unknown means the extracted words are put into file names with the following naming convention:

PREFIX_UNKNOWN_WORD_1.DIG,

PREFIX_UNKNOWN_WORD_2.DIG,

... ,

PREFIX_UNKNOWN_WORD_#.DIG.

Item 7 returns back the user to the extraction subsystem menu.

Database Operations

Item 2 from the main menu, brings the user to the database subsystem, who's menu shown on the next page.

Item 1 is used to add word names to the database. New word names are added to the end of the list. No sorting capability is provided. It is assumed that a user usually adds many words at a time. The adding words program prompts the user for that name of the word, then prompts the user to ask if another word is to be added. This iterative routine continues until the user enters an N.

```
1 - ADD WORD(S) TO DATABASE
2 - DELETE WORDS FROM DATABASE
3 - LIST LIBRARY WORDS
4 - SET DIRECTORY NAME
5 - SET LIBRARY PREFIX
6 - SET LIST POINTER
7 - QUIT EXTRACTION OPERATIONS
CHOICE ->
```

Item 2 is used to delete one or more words. The words may be deleted from anywhere in the list and uses the same iterative loop that the add word program.

Item 3 just list the word names in the database list.

Item 4 allows the user to set the location pointer of the database. The default is DJA23:[TRATHBUN.SPEECH.DATABASE].

Item 5 allows the user to set a prefix for the words in the database that make up the library. There is no default value.

Item 7 is used to set a pointer between using the database list and sublist. The default value is list.

Item 8 returns the user back to the main menu.

Feature Operations

Item 3 from the main menu, brings the user to the feature subsystem, who's menu shown below:

```
1 - SET MODE - SINGLE OR BATCH (BATCH)
2 - SET PREFIX NAME
3 - CREATE ALL FEATURES
4 - TO QUIT
```

Item 1 is used to toggle between creating features for all the words in the database list or just a single word. The default value is batch.

Item 2 is used to set a prefix value for the feature program. The prefix values is used in both the single and batch modes.

Item 3 creates all eight features of the word names supplied. If the batch mode is set then the feature program gets one word name at a time from the database list, adds the prefix and creates the features. If the single mode is set then the feature program prompts the user for the word name, add the prefix, and creates the features.

Item 4 returns to the main menu.

Recognition Operations

Item 4 from the maim menu brings the user to the recognition subsystem, whose menu is shown below:

```
 1 - BRING FEATURE(S) ONLINE
 2 - INPUT FILE FOR RECOGNITION TEST
 3 - SET RECOGNIZER TYPE (TIME WARP)
 4 - RECOGNIZE USING LP POWER SPECTRUM
 5 - RECOGNIZE USING LP DELTA POWER SPECTRUM
 6 - RECOGNIZE USING LF CEPSTRUM COEFFICIENTS
 7 - RECOGNIZE USING LF DELTA CEPSTRUM COEFFICIENTS
 8 - RECOGNIZE USING MF CEPSTRUM COEFFICIENTS
 9 - RECOGNIZE USING MF DELTA CEPSTRUM COEFFICIENTS
10 - RECOGNIZE USING BL CEPSTRUM COEFFICIENTS
11 - RECOGNIZE USING BL DELTA CEPSTRUM COEFFICIENTS
12 - RECOGNIZE USING ALL FEATURES
13 - RECOGNIZE USING FUSED FEATURES
14 - QUIT RECOGNITION OPERATIONS
```

Item 1 is used to bring the features online. The library words must be in memory in ordered to be used for recognition. The user can bring all or anyone of the features online at a time. After selecting item 1, the user is prompt with ALL FEATURES?. A Y response will bring all features online. A N response and the user will be prompt for each feature individually. This item works with the library prefix set in the database subsystem.

Item 2 is used to designate the file to be recognized. The user must specify the prefix in the file name designation.

Item 3 is used to designate which type of recognizer will be used in the recognition test. The choices are: time_warp, two_dee, error.

Items 4 through 11 are used to run a recognition test using the designating feature.

Item 12 is used to run a recognition test with all features.

Item 13 is used to create new features from the existing features for a recognition test. This function prompts the user for the name of the existing feature, how many components in the that feature, and if the user wants to add another feature. After the user responds

N to the last prompt, the user is prompted for the name of the new feature. Then the new feature is created and the recognition test is executed.

Item 14 quits the recognition subsystem and returns the user to the main menu.

Fusion Operations

Item 5 from the main menu brings the user to the fusion operations subsystem, whose menu is show on the next page.

Item 1 is used to designate where the output from items 4 through 7 will go.

Item 2 is used to input a file name for output designated to go to a file.

Item 3 creates a sublist. This function works on the fusion results list, so item 8 must be run first. The user is prompted for the number of words to be selected for the sublist.

```
 1 - SET OUTPUT CHANNEL (SCREEN OR FILE)
 2 - SET FILE NAME FOR RESULTS
 3 - CREATE SUBLIST
 4 - OUTPUT RESULTS FOR A FEATURE
 5 - OUTPUT RESULTS FOR ALL FEATURES
 6 - OUTPUT SMALLEST DTW VALUE FOR A FEATURE
 7 - OUTPUT SMALLEST DTW VALUE FOR ALL FEATURES
 8 - PERFORM FUSION
 9 - FUSION RESULT
10 - CLEAR FUSION LIST
11 - QUIT
```

Item 4 is used to output the distance result list for a specific entry in the fusion list. The user is prompted for the entry name. Note the entry name in the fusion list includes the feature name and which recognizers was used to create the distance result list. The naming convention is the feature name, an underscore, and a two letter abbreviation of the recognizer. The abbreviations are: time_warp - TW, two_dee - 2D, and error - ER. An example of recognizing LPPS with time warp produces a fusion list entry name of LPPS_TW.

Item 5 outputs the distances list for all the entries in the fusion list.

Item 6 outputs the word name and distance value of the recognition result with the smallest distance value, for a designated entry name in the fusion list.

Item 7 outputs the word name and distance value of the word with the smallest distance value for all entries in the fusion list.

Item 8 creates a new entry in the fusion list with the fusion results for all the entries. The entry name for the fusion results is FUS.

Item 9 outputs just the word name of the word in the FUS entry with the smallest recognition result.

Item 10 clears the fusion list. This must be done before the next recognition test or else the fusion list will be confused between which entries belong to which recognition attempt.

Item 11 exits the fusion subsystem and returns the user to the main menu.

Template Operations

Item 6 from the main menu bring the user to the Template Subsystem, whose menu is shown below:

```
1 - C, CREATE LIST OF PREFIXES
2 - C U, SET WORD NAME FOR TEMPLATING
3 - C U, SET OUTPUT TEMPLATE PREFIX
4 - U, SET TEMPLATE FACTOR
5 - U, SET INPUT TEMPLATE PREFIX
6 - CREATE TEMPLATE
7 - UPDATE TEMPLATE
8 - QUIT TEMPLATE SUBSYSTEM
CHOICE ->
```

Items 1 through 5 have either a C, C U, or U in front of them. This is an indicator to tell the user which item needs to be completed before creating or updating templates. The C refer to creating, the U refers to updating, and C U refers to both.

Item 1 is used to create a list of prefixes from which the template will be created. The user will be prompted for the prefix and whether to add another. The user enters an N to exit this item.

Item 2 is used to input the word name that the template will be created for.

Item 3 is used to input the prefix for the template. The user may create many templates for the same word, so the prefix is used to distinguish between template versions.

Item 4 is used to input the template factor. The template factor is a floating point value greater than 0.0.

Item 5 is used for naming the prefix of the template being updated. The output prefix for a template update will be the value from item 3. This value may be the same as item 5.

Item 6 is used to create templates. The user has the option of what features the template will be created for. The first prompt will ask the user if the templates should be created for all features. If the user responds N, then the user will be prompted for each feature.

Item 7 is used to update templates. The same prompting routines in item 6 is used for item 7.

Item 8 returns the user back to the main menu.

Creating Scripts

General rules about scripts:

- Line that begin with a "-' are comments and are ignored,

- The data for a command is always on the next line(s),

- Loop names must be unique,

- Loops exit when a word list finishes,

- All commands must be in upper case,

- unknown commands will be printed and execution stops.

PRINT_LINE

This command reads the next line and prints the contents to he screen.

```
PRINT_LINE
Any old statement
```

SET_DIRECTORY_NAME

This command reads the next line and set the contents to be the name of the directory for the database.

```
SET_DIRECTORY_NAME
[TRATHBUN.SPEECH.DATABASE]
```

SET_LIBRARY_PREFIX

This command reads the next line and set the contents to be prefix for the library words.

```
SET_LIBRARY_PREFIX
TOM1_
```

ADD_LIBRARY_WORD

This command reads the word name on the next line and adds it to the database list. The RESET_DATABASE_LIST command must be used after this command.

```
ADD_LIBRARY_WORD
PICKLES
```

```
RESET_DATABASE_LIST
```

DELETE_LIBRARY_WORD

This command reads the word name on the next line and deletes it from the database list. The RESET_DATABASE_LIST command must be used after this command.

```
DELETE_LIBRARY_WORD
PICKLES
RESET_DATABASE_LIST
```

RESET_DATABASE_LIST

This command rereads the word names for the database.dat file to create a new database list.

```
RESET_DATABASE_LIST
```

SET_LIST_ POINTER

This command reads the next line and sets it contents to be the list pointer. There are two possible values LIST and SUBLIST.

```
SET_LIST_POINTER
SUBLIST
```

RESET_WORD_LIST

This command sets the word pointer back to the beginning of the database list.

NEXT_WORD

This command loads the word_name variable with the word name pointed to by the word_pointer and increments the word_pointer to the next word name in the list.

SET_UNKNOWN_WORD_COUNT

This command reads the integer value on the next line and stores it into the unknown_word_count variable. The unknown_word_count is used with the next_unknown _word command for determine the next filename to generate.

```
SET_UNKNOWN_WORD_COUNT
1
```

NEXT_UNKNOWN_WORD

This command generates the next unknown word file name and loads this value into the word_name buffer. The names that are generated follow the naming convention UNKNOWN_WORD_#. The # is read from the unknown_word_name_count variable and is incremented after each use.

SET_FILE_DIRECTION

This commands reads the value of the next line and sets its to be the file_direction pointer. This pointer determines whether word names are input from the file or comes from the database list. The possible choices for this pointer are auto_create and script, with the former being used almost all the time.

```
SET_FILE_DIRECTION
AUTO_CREATE
```

SET_PREFIX

This command reads the value of the next line and stores this value into the pre_fix variable. This variable determines which set of words are used as the test words for recognition.

```
SET_PREFIX
TOM2_
```

SET_EXTRACTION_PREFIX

This command reads the value of the next line and stores it in the prefix variable in the EXTRACTION_PACKAGE. This command affects the file names of the words when they extracted using the extract command. The extract command must be used in the auto_unknown or auto_database mode.

```
SET_EXTRACTION_PREFIX
TOM3_
```

SET_EXTRACT_MODE

This command set the extraction mode to the value that follows. The possible choices are normal, auto_unknown, or auto_database. Normal cannot be used in a script command because this means the user inputs the file name as the words are extracted, this is not possible in a batch mode. Auto_unknown means the extraction program generates file names for the words that are extracted in the form UNKNOWN_WORD_#. Auto_database means that the extract program generates filename for the words begin extracted from the database list.

```
SET_EXTRACT_MODE
AUTO_UNKNOWN
```

THIS_WORD

This command display to the screen the value of the word_name variable. This is used to let the user known the progress of the script.

TRANSLATE

This command reads the file name on the next line and the input_channel value on the line after that, and uses them in the TRANSLATE program. When the input_channel is read, if the value is network, then the value of the next line is read to determine the network location where the file name is to be read from.

```
TRANSLATE
```

```
TOM1
NETWORK
CALVIN::DUA0:[TOMR]
```

MAKE_SPEECH_LIST

This command reads the file name on the next line and creates a speech list of it.

```
MAKE_SPEECH_LIST
TOM1
```

EXTRACT

This command automatically extracts words into a file. The make_speech_list or translate and create_feature_list commands must be executed first.

CREATE_FEATURE_LIST

This command uses the speech list and creates the feature list used for extraction. The make_speech_list or translate command must be executed first.

CREATE_ALL_FEATURE

This command creates the eight feature of the word name of the variable word_name when the file_direction is set to auto_create or the file name that follow the command when the file_direction is set to script.

SET_RECOGNIZER

This command reads the recognizer type on the next line set the recognizer type to be it. The recognizer types are time_warp, two_dee, error.

```
SET_RECOGNIZER
TIME_WARP
```

RECOGNIZE

This command executes the current recognizer for the a word name using the feature name on the next line. The word name is read from the word_name variable when the file_direction is set to auto_create. The word name is read from the line after the feature name when the file_direction is set to script. The feature names are the four letter acronyms LPPD, DPPS, LFCC, DFCC, MFMC, DFMC, BLMC, and DLMC.

```
RECOGNIZER
LPPS
```

UPDATE_WORD_LIST

This command reads the feature name on the next line and brings it online. All features to be used for recognition must be brought online first.

```
UPDATE_WORD_LIST
LPPS
```

FUSION_RECOGNITION

This command create fused features and executes the current recognizer. The number of values to follow this command varies depending on how many features are in the new super feature. The following three values are read in a looping fashion until the last one is N: name of the feature, number of components, and do you want to add another feature. After the last value is N, then the value on the next line is the name of the new feature. If the file direction is set to auto_create then the value of word_name is the file for recognition, if not then the filename must follow.

```
FUSION_RECOGNITION
LPPS
16
Y
DPPS
10
N
```

DL_PPS

ALL_FEATURE_RESULT_OUTPUT

This command outputs the feature results for all features to a file. The create_report_file command must be used first.

ALL_FEATURE_SMALLEST_OUTPUT

This command outputs the smallest feature results for all feature to a file. The create_report_file command must be used first.

FUSION

This command creates a fusion result for all the results in the fusion list.

FUSION_RESULT

This command displays to the screen the smallest value in the fusion results list. The fusion command must be executed first.

CREATE_SUBLIST

This command reads the integer value on the next line and creates a sublist of that size. The fusion command must be executed first.

CLEAR_FUSION_LIST

This command clears the fusion list, which must be done when going on to recognize the next word.

CREATE_REPORT_FILE

This command reads the file name on the next line and creates a report file of that file name.

```
CREATE_REPORT_FILE
TIME_WARP_RESULTS.DAT
```

CLOSE_OUTPUT_FILE

This command closes the report file.

CREATE_PREFIX_LIST

This command creates a prefix list from the data that follows for use with the create_tw_template command. The data to follow is read in a loop fashion consisting of two values: the prefix name and do you want to add another. When the latter value is N then the looping stops.

```
CREATE_PREFIX_LIST
TOM1_
Y
TOM2_
N
```

CREATE_TW_TEMPLATE

This command creates a template using the prefix list and output prefix value that is read from the next line. The output prefix value is the prefix value that is used to store the newly created word.

```
CREATE_TW_TEMPLATE
TOMS_
```

UPDATE_TEMPLATE

The command reads the next three lines to obtain the values of the out_prefix, the in_prefix, and the template_factor. The word name is gotten from the word_name variable.

```
UPDATE_TEMPLATE
TOMS_
TOM3_
3
```

IF_AGREE

This command is used as the IF part of an if-then-else statement. This command reads the floating point value from the next line to get the goal value. This goal value determines the percentage of recognition results that have to agree in order to make this statement true. The structure for the command is as follows:

```
IF_AGREE
0.9
true statements
ELSE
false statement
END_IF
```

It is not necessary to have any true statements, the ELSE may follow the goal value. Similarly, it is not necessary to have any false statements, the END_IF may follow the ELSE statement.

GOTO

This command is used to jump anywhere in the script. The destination name is read from the next line.

```
GOTO
TEMPLATE LOOP
```

LOOP

This command is used in conjunction with the goto command. The loop command means that the line that follows is a jump to label.

ELSE

This command is used in conjunction with the if_agree command. When if_agree is true and the interpreter reads this command it skips all lines until the end_if command. When if_agree is false then all lines are skipped until the ELSE command is read then execution starts again.

END_IF

This command is used in conjunction with the if_agree and else command. This command is a label for the else command to jump to. If this command is read otherwise then the interpreter does nothing and goes on to the next command.

END

This command tells the interpreter to stop executing commands and go back the main menu.

Apendix B Material and Equipment

The following material and equipment were used in this thesis:

- Digiatal Sound Corporation (DSC) analog to digital convertor.
- VAXStation II running the VMS operating system.
- VAX 6000 model 420 running the VMS operating system.
- VAX ADA compiler
- Shure Model SM54 noise-reducing microphone

Apendix C Computer Source Code

The following is a list of ADA source code developed for this thesis:

```
- [source_file_header_comment]
with TEXT_IO, STRING_OPERATIONS, CONTROL_C_INTERCEPTION, REC_SUPPORT_PACK-
AGE;
with LINKED_LIST, RECOGNITION_PACKAGE, FUSION_PACKAGE, FEATURE_PACKAGE;
with DATABASE_PACKAGE, FUSION_REC_PACKAGE, EXTRACTION_PACKAGE, TEM-
PLATE_PACKAGE;
use TEXT_IO, STRING_OPERATIONS, DATABASE_PACKAGE, FUSION_REC_PACKAGE;
use EXTRACTION_PACKAGE, FUSION_PACKAGE, RECOGNITION_PACKAGE, TEM-
PLATE_PACKAGE;
pragma ELABORATE (CONTROL_C_INTERCEPTION);
procedure SPEECH_CONTROL_PANEL is
- [procedure_header_comment]
   use FUSION_LIST, FUSED_FEATURE_LIST;
   type CHOICE_INTEGER is range 1..8;
   type INPUT_MODE_TYPE is (INTERACTIVE, BATCH);
   type FILE_DIRECTION_TYPE is (AUTO_CREATE, SCRIPT_FILE);

   INPUT_MODE : INPUT_MODE_TYPE;
   INPUT_FILE, REPORT_FILE : FILE_TYPE;
   PREFIX, IN_PREFIX, OUT_PREFIX, CARRIDGE_RETURN, INPUT_LINE, FILE_NAME :
VAR_STRING;
   FILE_DIRECTION : FILE_DIRECTION_TYPE := AUTO_CREATE;
   UNKNOWN_WORD_NAME, PRE_FIX, FEATURE_NAME, WORD_NAME : VAR_STRING;
   TO_LOOP, THE_FILE, NEW_FEATURE_NAME : VAR_STRING;
   RESULTS : RESULT_RECORD_TYPE;
   TEMP, UNKNOWN_WORD_COUNT, SUBLIST_SIZE, COMPONENTS : INTEGER := 1;
   TO_LIST : DATABASE_PACKAGE.LIST_TYPE;
   EXTRACT_MODE : EXTRACTION_PACKAGE.EXTRACT_MODE_TYPE;
   INPUT_CHANNEL : EXTRACTION_PACKAGE.INPUT_CHANNEL_TYPE;
   OUTPUT_CHANNEL : FUSION_REC_PACKAGE.OUTPUT_CHANNEL_TYPE := FU-
SION_REC_PACKAGE.FILE;
   NUMBER_CHAR, ONLINE, YES_OR_NO : CHARACTER;
   NUMBER_STRING : STRING(1..1);
   NEW_FEATURES : FUSION_REC_PACKAGE.FEATURE_RECORD_TYPE;
   FUSED_FEATURE_HEAD,FUSED_FEATURE_TAIL : FUSED_FEATURE_LIST.LIST;
   WORD_POINTER : DATABASE_LIST.LIST;
   UPDATE_FLAG, DOING_BATCH : BOOLEAN := FALSE;
   CHOICE : CHOICE_INTEGER;
   THIS_STRING : STRING(1..80);
   TEMP_FACTOR, GOAL_AGREE, PRECENT_AGREE : FLOAT;
   EXTRACTION_MODE : EXTRACTION_PACKAGE.EXTRACT_MODE_TYPE;

-  [later_declarative_item]...
begin
```

```
NEW_LINE (2);
PUT_LINE (ITEM => "Welcome to the AFIT Speech Recognition Test System");
NEW_LINE (2);
PUT_LINE (ITEM => "     _     ____   _____  __________");
PUT_LINE (ITEM => "    /\    /   \ |   \      |");
PUT_LINE (ITEM => "   /  \ |      ||     |    |");
PUT_LINE (ITEM => "  /    \ \       |  /      |");
PUT_LINE (ITEM => " |      | \__   |___/       |");
PUT_LINE (ITEM => " |_______|     \ | \        |");
PUT_LINE (ITEM => " |      |       \ | \       |");
PUT_LINE (ITEM => " |      ||     || \        |");
PUT_LINE (ITEM => " |      | \____/ |   \     |");
NEW_LINE (1);
DECLARE_LOOP: loop
  declare

    package FEATURES is new FEATURE_PACKAGE (NUMBER => 512,
    M => 20);

    use FEATURES;

    FEATURE_MODE : FEATURES.FEATURE_MODE_TYPE;
    RECOGNIZER : RECOGNITION_PACKAGE.RECOGNIZER_TYPE := RECOGNITION_PACK-
AGE.TIME_WARP;

    package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
    package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (FILE_DIREC-
TION_TYPE);
    package ENUMERATION_IO1 is new TEXT_IO.ENUMERATION_IO (EXTRACTION_PACK-
AGE.INPUT_CHANNEL_TYPE);
    package ENUMERATION_IO2 is new TEXT_IO.ENUMERATION_IO (EXTRACTION_PACK-
AGE.EXTRACT_MODE_TYPE);
    package ENUMERATION_IO3 is new TEXT_IO.ENUMERATION_IO (FEATURES.FEA-
TURE_MODE_TYPE);
    package ENUMERATION_IO4 is new TEXT_IO.ENUMERATION_IO (INPUT_MODE_TYPE);
    package ENUMERATION_IO5 is new TEXT_IO.ENUMERATION_IO (RECOGNITION_PACK-
AGE.RECOGNIZER_TYPE);
    package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
    package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
    use FLOAT_IO, ENUMERATION_IO, ENUMERATION_IO1;
    use ENUMERATION_IO2, ENUMERATION_IO3, CHOICE_INTEGER_IO, INTEGER_IO;
    use ENUMERATION_IO4, ENUMERATION_IO5;

    RESULTS_POINTER : FUSION_LIST.LIST := FUSION_HEAD;
```

```
begin
  MAIN: loop
begin
  if NOT DOING_BATCH then
NEW_LINE;
PUT_LINE (ITEM => "    1 - GOTO EXTRACTION OPERATIONS");
PUT_LINE (ITEM => "    2 - GOTO DATABASE OPERATIONS");
PUT_LINE (ITEM => "    3 - GOTO FEATURE OPERATIONS");
PUT_LINE (ITEM => "    4 - GOTO RECOGNITION OPERATIONS");
PUT_LINE (ITEM => "    5 - GOTO FUSION OPERATIONS");
PUT_LINE (ITEM => "    6 - GOTO TEMPLATE OPERATIONS");
PUT_LINE (ITEM => "    7 - GOTO BATCH OPERATIONS");
PUT_LINE (ITEM => "    8 - TO QUIT");
NEW_LINE;
PUT (ITEM => "    CHOICE -> ");
GET (ITEM => CHOICE);
GET_LINE(CARRIDGE_RETURN);
  else
CHOICE := 7;
  end if;
  case CHOICE is
when 1 =>
  EXTRACTION_CONTROL_PANEL;
when 2 =>
  DATABASE_CONTROL_PANEL(UPDATE_FLAG);
  exit MAIN when UPDATE_FLAG = TRUE;
when 3 =>
  FEATURE_CONTROL_PANEL;
when 4 =>
  RECOGNITION_CONTROL_PANEL;
when 5 =>
  FUSION_CONTROL_PANEL;
when 6 =>
  TEMPLATE_CONTROL_PANEL;
when 7 =>
  if NOT DOING_BATCH then
PUT_LINE (ITEM => "TYPE IN NAME OF SCRIPT FILE, LESS EXTENSION");
GET_LINE (ITEM => FILE_NAME);
FILE_NAME := FILE_NAME&".SCR";
OPEN (
FILE => INPUT_FILE,
MODE => IN_FILE,
NAME => FILE_NAME);
WORD_POINTER := DATABASE_HEAD;
DOING_BATCH := TRUE;
  end if;
  BEGIN

INNER: loop
  GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
  if STRING_VALUE(1,INPUT_LINE) = '-' then
NULL;
  elsif IS_EQUAL(INPUT_LINE,"PRINT_LINE") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
PUT_LINE (ITEM => INPUT_LINE);
  elsif IS_EQUAL(INPUT_LINE,"SET_DIRECTORY_NAME") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
SET_DIRECTORY_NAME(INPUT_LINE);
exit MAIN;
  elsif IS_EQUAL(INPUT_LINE,"SET_LIBRARY_PREFIX") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
SET_LIBRARY_PREFIX(INPUT_LINE);
exit MAIN;
  elsif IS_EQUAL(INPUT_LINE,"ADD_LIBRARY_WORD") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
ADD_LIBRARY_WORD(INPUT_LINE);
exit MAIN;
  elsif IS_EQUAL(INPUT_LINE,"DELETE_LIBRARY_WORD") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
DELETE_LIBRARY_WORD(INPUT_LINE);
exit MAIN;
  elsif IS_EQUAL(INPUT_LINE,"RESET_DATABASE_LIST") then
RESET_DATABASE_LIST;
  elsif IS_EQUAL(INPUT_LINE,"SET_LIST_POINTER") then
                    DATABASE_PACKAGE.ENUMERATION_IO.GET (FILE => INPUT_FILE,
ITEM => TO_LIST);
GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
SET_LIST_POINTER(TO_LIST);
  elsif IS_EQUAL(INPUT_LINE,"RESET_WORD_LIST") then
WORD_POINTER := DATABASE_HEAD;
  elsif IS_EQUAL(INPUT_LINE,"NEXT_WORD") then
NEXT_WORD(WORD_NAME,WORD_POINTER);
  elsif IS_EQUAL(INPUT_LINE,"SET_UNKNOWN_WORD_COUNT") then
```

```
                    GET (FILE => INPUT_FILE, ITEM => UNKNOWN_WORD_COUNT);
    GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
      elsif IS_EQUAL(INPUT_LINE,"NEXT_UNKNOWN_WORD") then
            if UNKNOWN_WORD_COUNT < 10 then
                      NUMBER_CHAR := CHARACTER'VAL(UN-
KNOWN_WORD_COUNT+48);
      WORD_NAME := "UNKNOWN_WORD_"&NUMBER_CHAR;
                    else
                      TEMP := UNKNOWN_WORD_COUNT/10;
      NUMBER_CHAR := CHARACTER'VAL(TEMP+48);
      WORD_NAME := "UNKNOWN_WORD_"&NUMBER_CHAR;
      TEMP := UNKNOWN_WORD_COUNT MOD 10;
      NUMBER_CHAR := CHARACTER'VAL(TEMP+48);
      WORD_NAME := WORD_NAME&NUMBER_CHAR;
                    end if;
    UNKNOWN_WORD_COUNT := UNKNOWN_WORD_COUNT + 1;
      elsif IS_EQUAL(INPUT_LINE,"SET_FILE_DIRECTION") then
    GET (FILE => INPUT_FILE, ITEM => FILE_DIRECTION);
    GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
      elsif IS_EQUAL(INPUT_LINE,"SET_PREFIX") then
    GET_LINE (FILE => INPUT_FILE, ITEM => PRE_FIX);
      elsif IS_EQUAL(INPUT_LINE,"SET_EXTRACTION_PREFIX") then
    GET_LINE (FILE => INPUT_FILE, ITEM => EXTRACTION_PACKAGE.PREFIX);
      elsif IS_EQUAL(INPUT_LINE,"SET_EXTRACT_MODE") then
    GET (FILE => INPUT_FILE, ITEM => EXTRACTION_MODE);
                    GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
    SET_EXTRACTION_MODE(EXTRACTION_MODE);
      elsif IS_EQUAL(INPUT_LINE,"THIS_WORD") then
        PUT (ITEM => "PROCESSING WORD -> ");

        PUT (ITEM => WORD_NAME);

    NEW_LINE;
      elsif IS_EQUAL(INPUT_LINE,"TRANSLATE") then
    GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
    GET (FILE => INPUT_FILE, ITEM => INPUT_CHANNEL);
    GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
                    if INPUT_CHANNEL = NETWORK then
                      GET_LINE (FILE => INPUT_FILE, ITEM => EXTRACTION_PACK-
AGE.DATABASE);
                    end if;
```

```
TRANSLATE (FILE_NAME,INPUT_CHANNEL);
  elsif IS_EQUAL(INPUT_LINE,"MAKE_SPEECH_LIST") then
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
MAKE_SPEECH_LIST (FILE_NAME);
  elsif IS_EQUAL(INPUT_LINE,"EXTRACT") then
GET (FILE => INPUT_FILE, ITEM => EXTRACT_MODE);
GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
EXTRACT (EXTRACT_MODE);
  elsif IS_EQUAL(INPUT_LINE,"CREATE_FEATURE_LIST") then
CREATE_EXTRACTION_FEATURE_LIST;
  elsif IS_EQUAL(INPUT_LINE,"CREATE_ALL_FEATURE") then
if FILE_DIRECTION = AUTO_CREATE  then
  FILE_NAME := PRE_FIX&WORD_NAME;
else
  GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
end if;
CREATE_ALL_FEATURE(FILE_NAME);
  elsif IS_EQUAL(INPUT_LINE,"SET_RECOGNIZER") then
                GET (FILE => INPUT_FILE, ITEM => RECOGNIZER);
                GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
  elsif IS_EQUAL(INPUT_LINE,"RECOGNIZE") then
GET_LINE (FILE => INPUT_FILE, ITEM => FEATURE_NAME);
if IS_EQUAL(FEATURE_NAME,"LPPS") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
      if RECOGNIZER = TIME_WARP then
RESULTS := LPPS_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LPPS_TW");
      elsif RECOGNIZER = ERROR then
RESULTS := LPPS_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LPPS_ER");
                  else
RESULTS := LPPS_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LPPS_2D");
  end if;
```

```
elsif IS_EQUAL(FEATURE_NAME,"DPPS") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
        if RECOGNIZER = TIME_WARP then
RESULTS := DPPS_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DPPS_TW");
        elsif RECOGNIZER = ERROR then
RESULTS := DPPS_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DPPS_ER");
                        else
RESULTS := DPPS_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DPPS_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"LFCC") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
        if RECOGNIZER = TIME_WARP then
RESULTS := LFCC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LFCC_TW");
        elsif RECOGNIZER = ERROR then
RESULTS := LFCC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LFCC_ER");
                        else
RESULTS := LFCC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"LFCC_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"DFCC") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
        if RECOGNIZER = TIME_WARP then
RESULTS := DFCC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFCC_TW");
        elsif RECOGNIZER = ERROR then
```

```
RESULTS := DFCC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFCC_ER");
                   else
RESULTS := DFCC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFCC_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"MFMC") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
      if RECOGNIZER = TIME_WARP then
RESULTS := MFMC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"MFMC_TW");
      elsif RECOGNIZER = ERROR then
RESULTS := MFMC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"MFMC_ER");
                   else
RESULTS := MFMC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"MFMC_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"DFMC") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
      if RECOGNIZER = TIME_WARP then
RESULTS := DFMC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFMC_TW");
      elsif RECOGNIZER = ERROR then
RESULTS := DFMC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFMC_ER");
                   else
RESULTS := DFMC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DFMC_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"BLMC") then
  if FILE_DIRECTION = AUTO_CREATE  then
```

```
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
        if RECOGNIZER = TIME_WARP then
RESULTS := BLMC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"BLMC_TW");
        elsif RECOGNIZER = ERROR then
RESULTS := BLMC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"BLMC_ER");
                    else
RESULTS := BLMC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"BLMC_2D");
  end if;
elsif IS_EQUAL(FEATURE_NAME,"DLMC") then
  if FILE_DIRECTION = AUTO_CREATE  then
FILE_NAME := PRE_FIX&WORD_NAME;
  else
GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
  end if;
        if RECOGNIZER = TIME_WARP then
RESULTS := DLMC_SPEECH.RECOGNIZE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DLMC_TW");
        elsif RECOGNIZER = ERROR then
RESULTS := DLMC_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DLMC_ER");
                    else
RESULTS := DLMC_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
ASSIGN(RESULTS.FEATURE_NAME,"DLMC_2D");
  end if;
end if;
ADD_TO(RESULTS,FUSION_HEAD,FUSION_TAIL);
  elsif IS_EQUAL(INPUT_LINE,"UPDATE_WORD_LIST") then
GET_LINE (FILE => INPUT_FILE, ITEM => FEATURE_NAME);
if IS_EQUAL(FEATURE_NAME,"LPPS") then
  LPPS_SPEECH.UPDATE_WORD_LIST;
elsif IS_EQUAL(FEATURE_NAME,"DPPS") then
  DPPS_SPEECH.UPDATE_WORD_LIST;
```

```
    elsif IS_EQUAL(FEATURE_NAME,"LFCC") then
      LFCC_SPEECH.UPDATE_WORD_LIST;
    elsif IS_EQUAL(FEATURE_NAME,"DFCC") then
      DFCC_SPEECH.UPDATE_WORD_LIST;
    elsif IS_EQUAL(FEATURE_NAME,"MFMC") then
      MFMC_SPEECH.UPDATE_WORD_LIST;
    elsif IS_EQUAL(FEATURE_NAME,"DFMC") then
      DFMC_SPEECH.UPDATE_WORD_LIST;
    elsif IS_EQUAL(FEATURE_NAME,"BLMC") then
      BLMC_SPEECH.UPDATE_WORD_LIST;
    elsif IS_EQUAL(FEATURE_NAME,"DLMC") then
      DLMC_SPEECH.UPDATE_WORD_LIST;
    end if;
      elsif IS_EQUAL(INPUT_LINE,"FUSION_RECOGNITION") then
    COMPONENTS := 0;
    CLEAR_LIST(FUSED_FEATURE_HEAD,FUSED_FEATURE_TAIL);
    INPUT: loop
      GET_LINE (FILE => INPUT_FILE, ITEM => NEW_FEATURES.NAME);
      GET (FILE => INPUT_FILE, ITEM => NEW_FEATURES.COMPONENTS);
      COMPONENTS := COMPONENTS + NEW_FEATURES.COMPONENTS;
      GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
       FUSED_FEATURE_LIST.ADD_TO(NEW_FEATURES,FUSED_FEA-
TURE_HEAD,FUSED_FEATURE_TAIL);
      GET (FILE => INPUT_FILE, ITEM => ONLINE);
      GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
    exit INPUT when ONLINE = 'N' OR ONLINE = 'n';
    end loop INPUT;
    GET_LINE (FILE => INPUT_FILE, ITEM => NEW_FEATURE_NAME);
    declare
      package FUSED_SPEECH is new REC_SUPPORT_PACKAGE
      (NUMBER_OF_COMPONENTS => COMPONENTS,
       NAME_OF_FEATURE => STRING_VALUE(NEW_FEATURE_NAME));
      use FUSED_SPEECH;
    begin
      if FILE_DIRECTION = AUTO_CREATE  then
    FILE_NAME := PRE_FIX&WORD_NAME;
      else
    GET_LINE (FILE => INPUT_FILE, ITEM => FILE_NAME);
      end if;
       FUSED_SPEECH.CREATE_FUSION_WORD_LIST(FUSED_FEATURE_HEAD,FUSED_FEA-
TURE_TAIL);
       FUSED_SPEECH.CREATE_FUSION_WORD(FILE_NAME,FUSED_FEA-
TURE_HEAD,FUSED_FEATURE_TAIL);
```

```
        if RECOGNIZER = TIME_WARP then
RESULTS := FUSED_SPEECH.RECOGNIZE(FILE_NAME);
RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_TW";
        elsif RECOGNIZER = ERROR then
RESULTS := FUSED_SPEECH.RECOGNIZE_ERROR(FILE_NAME);
RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_ER";
                    else
RESULTS := FUSED_SPEECH.RECOGNIZE_TWO_DEE(FILE_NAME);
RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_2D";
  end if;
  ADD_TO(RESULTS,FUSION_HEAD,FUSION_TAIL);
end;
  elsif IS_EQUAL(INPUT_LINE,"ALL_FEATURE_RESULT_OUTPUT") then
ALL_FEATURE_RESULTS_OUTPUT(OUTPUT_CHANNEL,REPORT_FILE);
  elsif IS_EQUAL(INPUT_LINE,"ALL_FEATURE_SMALLEST_RESULT") then
ALL_FEATURE_SMALLEST_RESULT(OUTPUT_CHANNEL,REPORT_FILE);
  elsif IS_EQUAL(INPUT_LINE,"FUSION") then
FUSION;
  elsif IS_EQUAL(INPUT_LINE,"FUSION_RESULT") then
FUSION_RESULT;
  elsif IS_EQUAL(INPUT_LINE,"CREATE_SUBLIST") then
                   INTEGER_IO.GET (FILE => INPUT_FILE, ITEM => SUBLIST_SIZE);
GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
CREATE_SUBLIST(SUBLIST_SIZE);
  elsif IS_EQUAL(INPUT_LINE,"CLEAR_FUSION_LIST") then
CLEAR_LIST(FUSION_HEAD,FUSION_TAIL);
  elsif IS_EQUAL(INPUT_LINE,"CREATE_REPORT_FILE") then
GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
CREATE (
  FILE => REPORT_FILE,
  MODE => OUT_FILE,
  NAME => INPUT_LINE);
  elsif IS_EQUAL(INPUT_LINE,"CLOSE_OUTPUT_FILE") then
CLOSE (FILE => REPORT_FILE);
  elsif IS_EQUAL(INPUT_LINE,"CREATE_PREFIX_LIST") then
PREFIX_LIST.CLEAR_LIST(PREFIX_HEAD,PREFIX_TAIL);
          GET_PREFIXES: loop
                    GET_LINE (FILE => INPUT_FILE, ITEM => PREFIX);
  PREFIX_LIST.ADD_TO(PREFIX,PREFIX_HEAD,PREFIX_TAIL);
                    GET (FILE => INPUT_FILE, ITEM => ONLINE);
  GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
exit GET_PREFIXES when ONLINE = 'N' OR ONLINE = 'n';
```

```
      end loop GET_PREFIXES;
    elsif IS_EQUAL(INPUT_LINE,"CREATE_TW_TEMPLATE") then
                        GET_LINE (FILE => INPUT_FILE, ITEM => OUT_PREFIX);
  LPPS_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  DPPS_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  LFCC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  DFCC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  MFMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  DFMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  BLMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
  DLMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,WORD_NAME);
    elsif IS_EQUAL(INPUT_LINE,"UPDATE_TEMPLATE") then
                        GET_LINE (FILE => INPUT_FILE, ITEM => OUT_PREFIX);
                        GET_LINE (FILE => INPUT_FILE, ITEM => IN_PREFIX);
                        GET (FILE => INPUT_FILE, ITEM => TEMP_FACTOR);
                        GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
  LPPS_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  DPPS_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  LFCC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  DFCC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  MFMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PRE-
FIX,WORD_NAME,TEMP_FACTOR);
  DFMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  BLMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,WORD_NAME,TEMP_FAC-
TOR);
  DLMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PRE-
FIX,WORD_NAME,TEMP_FACTOR);
    elsif IS_EQUAL(INPUT_LINE,"IF_AGREE") then
  PRECENT_AGREE := AGREE;
                        GET (FILE => INPUT_FILE, ITEM => GOAL_AGREE);
                        GET_LINE (FILE => INPUT_FILE, ITEM => CARRIDGE_RETURN);
      if PRECENT_AGREE < GOAL_AGREE then
                          TO_ELSE: loop
                            GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
          exit TO_ELSE when IS_EQUAL(INPUT_LINE,"ELSE");
                          end loop TO_ELSE;
                        end if;
    elsif IS_EQUAL(INPUT_LINE,"GOTO") then
                        GET_LINE (FILE => INPUT_FILE, ITEM => TO_LOOP);
  RESET(FILE => INPUT_FILE);
```

```
LOOPITY: loop
  GET_LINE (FILE => INPUT_FILE, ITEM => INPUT_LINE);
  exit LOOPITY when IS_EQUAL(INPUT_LINE,TO_LOOP);
end loop LOOPITY;
  elsif IS_EQUAL(INPUT_LINE,"LOOP") then
                GET_LINE (FILE => INPUT_FILE, ITEM => TO_LOOP);
  elsif IS_EQUAL(INPUT_LINE,"ELSE") then
TO_END_IF: loop
  GET_LINE (INPUT_FILE, ITEM => INPUT_LINE);
exit TO_END_IF when IS_EQUAL(INPUT_LINE,"END_IF");
end loop TO_END_IF;
  elsif IS_EQUAL(INPUT_LINE,"END_IF") then
NULL;
  elsif IS_EQUAL(INPUT_LINE,"END") then
CLOSE (FILE => INPUT_FILE);
DOING_BATCH := FALSE;
exit MAIN;
  else
PUT_LINE (ITEM => "THE FOLLOW COMMAND WAS NOT RECONIZED");
PUT_LINE (ITEM => INPUT_LINE);
DOING_BATCH := FALSE;
exit DECLARE_LOOP;
  end if;
end loop INNER;
  exception
when DATABASE_PACKAGE.DATABASE_LIST.UNDER_FLOW | NAME_ERROR =>
  TO_END_OF_LOOP: loop
GET_LINE (INPUT_FILE, ITEM => INPUT_LINE);
  exit TO_END_OF_LOOP when IS_EQUAL(INPUT_LINE,"GOTO");
  end loop TO_END_OF_LOOP;
  GET_LINE (INPUT_FILE, ITEM => INPUT_LINE);
  end;
when 8 =>
  exit DECLARE_LOOP;
end case;
  exception
when DATA_ERROR =>
  NEW_LINE;
  PUT_LINE (ITEM => "FOR CRYING OUT LOUD, PLEASE TYPE A NUMBER BETWEEN 1
AND 6, SHEEEEEESH");
  end;
end loop MAIN;
```

```
        end;
    end loop DECLARE_LOOP;
    NEW_LINE(2);
    PUT_LINE (ITEM => "BYE BYE, PLEASE COME AGAIN");
-[exception_part]
end SPEECH_CONTROL_PANEL;
```

```
- [source_file_header_comment]
with TEXT_IO, SEQUENTIAL_IO, FLOAT_MATH_LIB, STRING_OPERATIONS;
with LINKED_LIST, DATABASE_PACKAGE, FEATURE_PACKAGE;
use TEXT_IO, FLOAT_MATH_LIB, STRING_OPERATIONS, DATABASE_PACKAGE;
package EXTRACTION_PACKAGE IS
- [package_specification_header_comment]
  type HEADER_TYPE is array (INTEGER RANGE 1..256) of SHORT_INTEGER;
  package IN_INTEGER_IO is new SEQUENTIAL_IO (SHORT_INTEGER);
  package IN2_INTEGER_IO is new SEQUENTIAL_IO (HEADER_TYPE);

  type EXTRACT_MODE_TYPE is (NORMAL, AUTO_UNKNOWN, AUTO_DATABASE);
  type INPUT_CHANNEL_TYPE is (NETWORK, LOCAL);
  INPUT_CHANNEL : INPUT_CHANNEL_TYPE := NETWORK;
  VARIABLE : HEADER_TYPE;
  EXTRACT_MODE : EXTRACT_MODE_TYPE := NORMAL;
  INPUT_VARIABLE : SHORT_INTEGER;
  OUTPUT_VARIABLE : FLOAT;
  INPUT_FILE : IN2_INTEGER_IO.FILE_TYPE;
  SPEECH_FILE, OUTPUT_FILE : TEXT_IO.FILE_TYPE;
  DUMMY, EXTENSION, FILE, FILE_NAME, FILE_NAME_TOO : STRING_OPERA-
TIONS.VAR_STRING;
  POINTER, COUNT : INTEGER;
  START_COUNT : INTEGER := 6;
  END_COUNT : INTEGER := 12;
  FILTER : INTEGER := 5;
  PREFIX, DATABASE : VAR_STRING;
  CHANGE : CHARACTER;

  package FEATURE_LIST is new LINKED_LIST (INTEGER);
  package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
  package OUT_FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
  package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (INPUT_CHANNEL_TYPE);
  package EX_ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (EX-
TRACT_MODE_TYPE);
  use OUT_FLOAT_IO, INTEGER_IO, ENUMERATION_IO;
  use FEATURE_LIST, EX_ENUMERATION_IO;

  FEATURE_HEAD, FEATURE_TAIL : FEATURE_LIST.LIST;

  - [procedure_header_comment]
  procedure TRANSLATE (FILE_NAME : in VAR_STRING;
    INPUT_CHANNEL : in INPUT_CHANNEL_TYPE);
  - [procedure_header_comment]
  procedure MAKE_SPEECH_LIST (FILE_NAME : in VAR_STRING);
  - [procedure_header_comment]
  procedure EXTRACT (EXTRACTION_MODE : in EXTRACT_MODE_TYPE);
  - [procedure_header_comment]
  procedure REVIEW_EXTRACTION;
  - [procedure_header_comment]
  procedure CREATE_EXTRACTION_FEATURE_LIST;
  - [procedure_header_comment]
  procedure EXTRACTION_CONTROL_PANEL;

  - [procedure_header_comment]
  procedure SET_EXTRACTION_MODE (EXTRACTION_MODE : in EXTRACT_MODE_TYPE);

end EXTRACTION_PACKAGE;
```

```
package body EXTRACTION_PACKAGE is
- [package_body_header_comment]

  MEAN, STAN_DEV : FLOAT;
  NUMBER : INTEGER := 256;
  COEFFICIENT : FLOAT := 2.0;
  package FEATURES is new FEATURE_PACKAGE (NUMBER => NUMBER, M => 20);
  use FEATURES;

  FEATURE_POINTER : FEATURE_LIST.LIST := FEATURE_HEAD;
  AVERAGE_ARRAY : FEATURE_ARRAY_TYPE(1..50);
  AVERAGE_POINTER : INTEGER RANGE 1..FILTER := 1;

  function MOVING_AVERAGE (NEW_DATUM : in FLOAT) return FLOAT is
  - [function_header_comment]
    SUM : FLOAT := 0.0;
  begin
    AVERAGE_ARRAY(AVERAGE_POINTER) := NEW_DATUM;
    for INDEX in 1..FILTER loop
      SUM := SUM + AVERAGE_ARRAY(INDEX);
    end loop;
    if AVERAGE_POINTER = FILTER then
      AVERAGE_POINTER := 1;
    else
      AVERAGE_POINTER := AVERAGE_POINTER + 1;
    end if;
    return SUM / FLOAT(FILTER);
  end MOVING_AVERAGE;

  procedure TRANSLATE (FILE_NAME : in VAR_STRING;
     INPUT_CHANNEL : in INPUT_CHANNEL_TYPE) is
  - [procedure_header_comment]
    LAST : FLOAT := 0.0;
    FULL_FILE_NAME : VAR_STRING;
  begin

    - CONQUER FILE
    SPEECH_LIST.CLEAR_LIST(SPEECH_HEAD,SPEECH_TAIL);
    ASSIGN(EXTENSION,".SND");
    if INPUT_CHANNEL = LOCAL then
      FULL_FILE_NAME := DIRECTORY_NAME&FILE_NAME&EXTENSION;
          else
    FULL_FILE_NAME := DATABASE&FILE_NAME&EXTENSION;
  end if;
  IN2_INTEGER_IO.OPEN (
    FILE => INPUT_FILE,
    MODE => IN2_INTEGER_IO.IN_FILE,
    NAME => STRING_VALUE(FULL_FILE_NAME));
  ASSIGN(EXTENSION,".DIG");
  FULL_FILE_NAME := DIRECTORY_NAME&FILE_NAME&EXTENSION;
  CREATE (
    FILE => OUTPUT_FILE,
    MODE => OUT_FILE,
    NAME => FULL_FILE_NAME);
  IN2_INTEGER_IO.READ (FILE => INPUT_FILE, ITEM => VARIABLE(1..256));
  LOOP
    IN2_INTEGER_IO.READ(FILE => INPUT_FILE, ITEM => VARIABLE(1..256));
    for INDEX in 1..256 loop
  OUTPUT_VARIABLE := FLOAT(VARIABLE(INDEX));
  PUT (FILE => OUTPUT_FILE, ITEM => OUTPUT_VARIABLE,
    FORE => 6,
    AFT => 2,
    EXP => 0);
  NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
  SPEECH_LIST.ADD_TO(OUTPUT_VARIABLE,SPEECH_HEAD,SPEECH_TAIL);
    end loop;
  end loop;
exception
  when END_ERROR =>
    IN2_INTEGER_IO.CLOSE (FILE => INPUT_FILE);
    CLOSE (FILE => OUTPUT_FILE);
end TRANSLATE;

procedure SET_EXTRACTION_MODE (EXTRACTION_MODE : in EXTRACT_MODE_TYPE) is
- [procedure_header_comment]
begin
  EXTRACT_MODE := EXTRACTION_MODE;
end SET_EXTRACTION_MODE;

procedure MAKE_SPEECH_LIST (FILE_NAME : in VAR_STRING)is
- [procedure_header_comment]
  FULL_FILE_NAME : VAR_STRING;
  SPEECH_DATUM : FLOAT;
  INPUT_FILE : TEXT_IO.FILE_TYPE;
```

```
begin
  SPEECH_LIST.CLEAR_LIST(SPEECH_HEAD,SPEECH_TAIL);
  ASSIGN(EXTENSION,".DIG");
  FULL_FILE_NAME := DIRECTORY_NAME&FILE_NAME&EXTENSION;
  OPEN (
    FILE => INPUT_FILE,
    MODE => IN_FILE,
    NAME => FULL_FILE_NAME);
  loop
    GET (FILE => INPUT_FILE, ITEM => SPEECH_DATUM);
    SPEECH_LIST.ADD_TO(SPEECH_DATUM,SPEECH_HEAD,SPEECH_TAIL);
  end loop;
exception
  when END_ERROR =>
    CLOSE (FILE => INPUT_FILE);
end MAKE_SPEECH_LIST;


procedure CREATE_EXTRACTION_FEATURE_LIST is
- [procedure_header_comment]

  type FEATURE_TYPES is array (1..16) of FLOAT;
  DISTANCE_FACTOR : INTEGER;
  SPECTRUM, SPECTRUM_TEST : FEATURE_ARRAY_TYPE(1..16);
  THRESHOLD, DISTANCE, NORMALIZE : FLOAT := 0.0;
  ZC_AVE_ARRAY, RAW_SPEECH : FEATURE_ARRAY_TYPE(1..NUMBER);
  SPEECH_POINTER : SPEECH_LIST.LIST := SPEECH_HEAD;

  function EUCLIDIAN_DISTANCE (VECTOR1, VECTOR2 : in FEATURE_ARRAY_TYPE) return
FLOAT is
  - [function_header_comment]
    SUM : FLOAT := 0.0;
  begin
    for INDEX in 1..16 loop
  SUM := SUM + (VECTOR1(INDEX) - VECTOR2(INDEX))**2;
    end loop;
    return SQRT(SUM);
  end EUCLIDIAN_DISTANCE;

  function FIND_MEAN (DATAS : in FEATURE_ARRAY_TYPE) return FLOAT is
  - [function_header_comment]
```

```
      SUM : FLOAT := 0.0;
    begin
      for INDEX in 1..NUMBER loop
        SUM := SUM + ABS(DATAS(INDEX));
      end loop;
      return SUM/FLOAT(NUMBER);
    end FIND_MEAN;

    function FIND_STAN_DEV (DATAS : in FEATURE_ARRAY_TYPE) return FLOAT is
    - [function_header_comment]
      SUM : FLOAT := 0.0;
    begin
      for INDEX in 1..NUMBER loop
        SUM := SUM + (ABS(DATAS(INDEX)) - MEAN)**2;
      end loop;
      return SQRT(SUM/FLOAT(NUMBER));
    end FIND_STAN_DEV;
  begin
    FEATURE_LIST.CLEAR_LIST(FEATURE_HEAD,FEATURE_TAIL);
    for INDEX in 1..NUMBER loop
      RAW_SPEECH(INDEX) := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
      SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
    end loop;
    MEAN := FIND_MEAN(RAW_SPEECH);
    STAN_DEV := FIND_STAN_DEV(RAW_SPEECH);
    FEATURES.POWER_SPECTRUM(RAW_SPEECH,SPECTRUM_TEST);
     ADD_TO(0,FEATURE_HEAD,FEATURE_TAIL);
    for INDEX1 in 1..9 loop
      for INDEX in 1..NUMBER loop
    RAW_SPEECH(INDEX) := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
    SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
      end loop;
      FEATURES.POWER_SPECTRUM(RAW_SPEECH,SPECTRUM);
       ADD_TO(0,FEATURE_HEAD,FEATURE_TAIL);
      DISTANCE := DISTANCE + EUCLIDIAN_DISTANCE(SPECTRUM,SPECTRUM_TEST);
    end loop;
    THRESHOLD := DISTANCE / 9.0;
    loop
      for INDEX in 1..NUMBER loop
    RAW_SPEECH(INDEX) := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
    SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
      end loop;
      FEATURES.POWER_SPECTRUM(RAW_SPEECH,SPECTRUM);
      DISTANCE := EUCLIDIAN_DISTANCE(SPECTRUM,SPECTRUM_TEST);
      DISTANCE_FACTOR := INTEGER(DISTANCE/THRESHOLD);
      if DISTANCE_FACTOR > 9 then
         DISTANCE_FACTOR := 9;
      end if;
      ADD_TO(DISTANCE_FACTOR,FEATURE_HEAD,FEATURE_TAIL);
   end loop;
exception
   when SPEECH_LIST.UNDER_FLOW =>
      FEATURE_POINTER := FEATURE_HEAD;
end CREATE_EXTRACTION_FEATURE_LIST;

procedure DISPLAY_FEATURE_LIST (NUMBER_OF_LINES : in INTEGER := -1) is
- [procedure_header_comment]
   POINTER : FEATURE_LIST.LIST := FEATURE_POINTER;
   LINES, VALUE, COUNTER : INTEGER := 1;
begin
   for INDEX in 1..80 loop
      PUT (ITEM => INDEX / 10, WIDTH => 1);
   end loop;
   NEW_LINE;
   for INDEX in 1..80 loop
      PUT (ITEM => INDEX MOD 10, WIDTH => 1);
   end loop;
   NEW_LINE;
   PUT_LOOP: loop
      if COUNTER = 81 then
   NEW_LINE;
   COUNTER := 1;
   LINES := LINES + 1;
      end if;
   exit PUT_LOOP when NUMBER_OF_LINES + 1 = LINES;
      VALUE := GET_NODE(POINTER);
      PUT (ITEM => VALUE, WIDTH => 1);
      COUNTER := COUNTER + 1;
      POINTER := NEXT_LINK(POINTER);
   end loop PUT_LOOP;
exception
   when FEATURE_LIST.UNDER_FLOW =>
      NEW_LINE;
end DISPLAY_FEATURE_LIST;
```

```
  procedure ADVANCE_POINTER (POINTER : in out SPEECH_LIST.LIST; NUMBER : in INTEGER)
is
  - [procedure_header_comment]
  begin
    for INDEX in 1..NUMBER loop
      POINTER := SPEECH_LIST.NEXT_LINK(POINTER);
    end loop;
  end ADVANCE_POINTER;

  procedure EXTRACT (EXTRACTION_MODE : in EXTRACT_MODE_TYPE) is
  - [procedure_header_comment]
     FULL_FILE_NAME : VAR_STRING;
    SPEECH_POINTER : SPEECH_LIST.LIST := SPEECH_HEAD;
    FEATURE_POINTER : FEATURE_LIST.LIST := FEATURE_HEAD;
    UNKNOWN_WORD_COUNT, END_COUNTER, START_COUNTER : INTEGER := 0;
    TEMP, FEATURE_DATUM : INTEGER;
    SPEECH_DATUM : FLOAT;
    WORD_IN_PROGRESS : BOOLEAN := FALSE;
    UNKNOWN_FILE : STRING(1..15);
    WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;

  begin
    loop
      FEATURE_DATUM := GET_NODE(FEATURE_POINTER);
      FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
      if FEATURE_DATUM > 2 then
    START_COUNTER := 1;
        START_WORD: loop
      FEATURE_DATUM := GET_NODE(FEATURE_POINTER);
      FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
    exit START_WORD when FEATURE_DATUM < 3;
      START_COUNTER := START_COUNTER + 1;
        end loop START_WORD;
        if START_COUNTER >= START_COUNT then
      WORD_IN_PROGRESS := TRUE;
          EX_WORD: loop
    END_COUNTER := 1;
    UNDER_WORD: loop
      FEATURE_DATUM := GET_NODE(FEATURE_POINTER);
      FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
    exit UNDER_WORD when FEATURE_DATUM > 2 OR END_COUNTER = END_COUNT;
```

```
      END_COUNTER := END_COUNTER + 1;
    end loop UNDER_WORD;
             if END_COUNTER = END_COUNT then
               if EXTRACTION_MODE = NORMAL then
    TEXT_IO.PUT_LINE (ITEM => "WORD HAS BEEN FOUND, TYPE IN FILE NAME, LESS EX-
TENSION");
    STRING_OPERATIONS.GET_LINE(FILE);
               elsif EXTRACTION_MODE = AUTO_DATABASE then
                 NEXT_WORD(FILE,WORD_POINTER);
               else
    UNKNOWN_WORD_COUNT := UNKNOWN_WORD_COUNT + 1;
    UNKNOWN_FILE(1..13) := "UNKNOWN_WORD_";
                 if UNKNOWN_WORD_COUNT < 10 then
      UNKNOWN_FILE(14) := CHARACTER'VAL(UNKNOWN_WORD_COUNT+48);
                   ASSIGN (FILE,UNKNOWN_FILE(1..14));
                 else
      TEMP := UNKNOWN_WORD_COUNT/10;
          for INDEX in 0..1 loop
        UNKNOWN_FILE(14+INDEX) := CHARACTER'VAL(TEMP+48);
    TEMP := UNKNOWN_WORD_COUNT MOD 10;
                   end loop;
                   ASSIGN(FILE,UNKNOWN_FILE(1..14));
                 end if;
               end if;
      FULL_FILE_NAME := DIRECTORY_NAME&PREFIX&FILE&".DIG";
      STRING_OPERATIONS.CREATE (
     FILE => OUTPUT_FILE,
     MODE => OUT_FILE,
     NAME => FULL_FILE_NAME);
      for INDEX in 1..NUMBER*(START_COUNTER) loop
    SPEECH_DATUM := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
    PUT (FILE => OUTPUT_FILE, ITEM => SPEECH_DATUM,
      FORE => 6,
      AFT => 2,
      EXP => 0);
    TEXT_IO.NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
    SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
      end loop;
      CLOSE (FILE => OUTPUT_FILE);
               for INDEX in 1..END_COUNT loop
                 ADVANCE_POINTER(SPEECH_POINTER,NUMBER);
                        end loop;
          WORD_IN_PROGRESS := FALSE;
        else
          START_COUNTER := START_COUNTER + END_COUNTER + 1;
          WORD: loop
        FEATURE_DATUM := GET_NODE(FEATURE_POINTER);
        FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
          exit WORD when FEATURE_DATUM < 3;
        START_COUNTER := START_COUNTER + 1;
          end loop WORD;
                end if;
              exit EX_WORD when WORD_IN_PROGRESS = FALSE;
              end loop EX_WORD;
            else
              for INDEX in 1..START_COUNTER+1 loop
        ADVANCE_POINTER(SPEECH_POINTER,NUMBER);
              end loop;
            end if;
          else
            ADVANCE_POINTER(SPEECH_POINTER,NUMBER);
          end if;
        end loop;
      exception
        when DATABASE_LIST.UNDER_FLOW =>
          TEXT_IO.PUT_LINE (ITEM => "THE DATABASE LIST HAS RUN OUT OF WORDS");
        when FEATURE_LIST.UNDER_FLOW =>
          if WORD_IN_PROGRESS then
        TEXT_IO.PUT_LINE (ITEM => "THE LAST WORD WAS NOT FULLY EXTRACTED, RERE-
    CORD IT");
          end if;
      end EXTRACT;

      procedure MANUAL_EXTRACTION is
      - [procedure_header_comment]
        subtype CHOICE_TYPE is INTEGER range 1..9;

        AVERAGE : FLOAT;
        ZC_COUNT : INTEGER;
        FILE, FILE_NAME : VAR_STRING;
        OUTPUT_FILE : FILE_TYPE;
        OUT_PUT : BOOLEAN;
        CHOICE : CHOICE_TYPE;
        SKIP_NUM, SKIP_COUNT, NUM_COUNT, VALUE : INTEGER;
```

```
ZC_THRESHOLD, THRESHOLD, SPEECH_DATUM : FLOAT;
BACK_SPEECH_POINTER, SPEECH_POINTER : SPEECH_LIST.LIST := SPEECH_HEAD;
BACK_FEATURE_POINTER : FEATURE_LIST.LIST := FEATURE_POINTER;

package TEMP_LIST is new LINKED_LIST (ITEM_TYPE => FLOAT);
package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_TYPE);
use CHOICE_INTEGER_IO,TEMP_LIST;

TEMP_HEAD, TEMP_TAIL : TEMP_LIST.LIST;

begin
 REVIEW_LOOP: loop
   begin
     NEW_LINE(2);
 DISPLAY_FEATURE_LIST(3);
     NEW_LINE(2);
 PUT_LINE (ITEM => "   1 - TO DISPLAY THE WHOLE FILE");
 PUT_LINE (ITEM => "   2 - TO DELETE A SECTION");
 PUT_LINE (ITEM => "   3 - TO WRITE A SECTION TO A FILE WITH TRIMMING");
 PUT_LINE (ITEM => "   4 - TO WRITE A SECTION TO A FILE WITHOUT TRIMMING");
 PUT_LINE (ITEM => "   5 - TO UNDO LAST WRITE");
 PUT_LINE (ITEM => "   6 - TO CHANGE COEFFICIENT VALUE");
 PUT_LINE (ITEM => "   7 - TO CHANGE THE FILTER SIZE");
 PUT_LINE (ITEM => "   8 - TO GOTO BEGINNING OF LIST");
 PUT_LINE (ITEM => "   9 - TO EXIT");
 NEW_LINE;
 PUT ("   CHOICE -> ");
     CHOICE_INTEGER_IO.GET (ITEM => CHOICE);
     GET_LINE (ITEM => DUMMY);
     case CHOICE is
       when 1 =>
 DISPLAY_FEATURE_LIST;
       when 2 =>
         PUT (ITEM => "NUMBER OF TIME SLICES: ");
 INTEGER_IO.GET(NUM_COUNT);
 GET_LINE(DUMMY);
         for INDEX in 1..NUM_COUNT loop
   FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
   ADVANCE_POINTER(SPEECH_POINTER,NUMBER);
         end loop;
       when 3 =>
```

```
    BACK_SPEECH_POINTER := SPEECH_POINTER;
    BACK_FEATURE_POINTER := FEATURE_POINTER;
        PUT_LINE (ITEM => "NAME OF FILE, LESS EXTENSION");
        GET_LINE (ITEM => FILE);
            PUT (ITEM => "NUMBER OF TIME SLICES: ");
    INTEGER_IO.GET(NUM_COUNT);
    GET_LINE(DUMMY);
    FILE_NAME := DIRECTORY_NAME&PREFIX&FILE&".DIG";
            CREATE (
              FILE => OUTPUT_FILE,
              MODE => OUT_FILE,
              NAME => FILE_NAME);

    AVERAGE_ARRAY := (OTHERS => 0.0);
    AVERAGE_POINTER := 1;
    THRESHOLD := MEAN + COEFFICIENT*STAN_DEV;
    OUT_PUT := FALSE;
    SKIP_COUNT := 1;
    SKIP_NUM := 0;
            SKIP_LOOP: loop
              exit SKIP_LOOP when OUT_PUT or SKIP_COUNT > NUM_COUNT or SKIP_NUM =
1024;
      for INDEX1 in 1..NUMBER loop
    SPEECH_DATUM := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
    SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
    AVERAGE := MOVING_AVERAGE(ABS(SPEECH_DATUM));
    if AVERAGE > THRESHOLD OR OUT_PUT then
      OUT_PUT := TRUE;
      ADD_TO_REVERSE(SPEECH_DATUM,TEMP_HEAD,TEMP_TAIL);
            else
      SKIP_NUM := SKIP_NUM + 1;
    end if;
      end loop;
      FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
              SKIP_COUNT := SKIP_COUNT + 1;
            end loop SKIP_LOOP;
            PUT (ITEM => "NUMBER OF POINTS DELETED IS ");
    INTEGER_IO.PUT (ITEM => SKIP_NUM, WIDTH => 4);
    NEW_LINE;
            for INDEX in SKIP_COUNT..NUM_COUNT loop
              for INDEX1 in 1..NUMBER loop
                SPEECH_DATUM := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
```

```
SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
ADD_TO_REVERSE(SPEECH_DATUM,TEMP_HEAD,TEMP_TAIL);
          end loop;
  FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
        end loop;

AVERAGE_ARRAY := (OTHERS => 0.0);
AVERAGE_POINTER := 1;
SKIP_NUM := 0;
        TAIL_SHAVING: loop
          REMOVE_NEXT_NODE(SPEECH_DATUM,TEMP_HEAD,TEMP_TAIL);
  AVERAGE := MOVING_AVERAGE(ABS(SPEECH_DATUM));
        exit TAIL_SHAVING when AVERAGE > THRESHOLD OR SKIP_NUM = 1024;
  SKIP_NUM := SKIP_NUM + 1;
        end loop TAIL_SHAVING;
        PUT (ITEM => "NUMBER OF POINTS DELETED IS ");
INTEGER_IO.PUT (ITEM => SKIP_NUM, WIDTH => 4);
NEW_LINE;
REVERSE_LIST(TEMP_HEAD,TE    _TAIL);
        begin
  OUTPUT: loop
REMOVE_NEXT_NODE(SPEECH_DATUM,TEMP_HEAD,TEMP_TAIL);
PUT (FILE => OUTPUT_FILE, ITEM => SPEECH_DATUM,
  FORE => 8,
  AFT => 2,
  EXP => 0);
NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
  end loop OUTPUT;
        exception
          when TEMP_LIST.UNDER_FLOW =>
CLOSE (FILE => OUTPUT_FILE);
        end;
      when 4 =>

BACK_SPEECH_POINTER := SPEECH_POINTER;
BACK_FEATURE_POINTER := FEATURE_POINTER;
    PUT_LINE (ITEM => "NAME OF FILE, LESS EXTENSION");
    GET_LINE (ITEM => FILE);
        PUT (ITEM => "NUMBER OF TIME SLICES: ");
INTEGER_IO.GET(NUM_COUNT);
GET_LINE(DUMMY);
FILE_NAME := DIRECTORY_NAME&PREFIX&FILE&".DIG";
```

```
              CREATE (
                FILE => OUTPUT_FILE,
                MODE => OUT_FILE,
                NAME => FILE_NAME);

      for INDEX in 1..NUM_COUNT loop
        for INDEX1 in 1..NUMBER loop
      SPEECH_DATUM := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
      SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
      PUT (FILE => OUTPUT_FILE, ITEM => SPEECH_DATUM,
        FORE => 8,
        AFT => 2,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
        end loop;
        FEATURE_POINTER := NEXT_LINK(FEATURE_POINTER);
      end loop;
      CLOSE (FILE => OUTPUT_FILE);
            when 5 =>
      SPEECH_POINTER := BACK_SPEECH_POINTER;
      FEATURE_POINTER := BACK_FEATURE_POINTER;
            when 6 =>
              PUT (ITEM => "NEW VALUE: ");
      GET(COEFFICIENT);
      GET_LINE(DUMMY);
            when 7 =>
              PUT (ITEM => "NEW VALUE: ");
      INTEGER_IO.GET(FILTER);
      GET_LINE(DUMMY);
            when 8 =>
      FEATURE_POINTER := FEATURE_HEAD;
      SPEECH_POINTER := SPEECH_HEAD;
            when 9 =>
              exit REVIEW_LOOP;
          end case;
        exception
          when DATA_ERROR =>
            PUT_LINE (ITEM => "WATCH IT PAL");
        end;
     end loop REVIEW_LOOP;
    end MANUAL_EXTRACTION;
```

```
procedure REVIEW_EXTRACTION is
- [procedure_header_comment]
  subtype CHOICE_TYPE is INTEGER range 1..9;
  CHOICE : CHOICE_TYPE;
  package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_TYPE);
  package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
  use CHOICE_INTEGER_IO, FLOAT_IO;
begin
  REVIEW_LOOP: loop
    begin
      NEW_LINE(2);
  PUT (ITEM => "  1 - THE BEGINNING WORD COUNT IS ");
  INTEGER_IO.PUT (ITEM => START_COUNT, WIDTH => 6);
  NEW_LINE;
  PUT (ITEM => "  2 - THE END WORD COUNT IS ");
  INTEGER_IO.PUT (ITEM => END_COUNT, WIDTH => 6);
  NEW_LINE;
  PUT (ITEM => "  3 - THE SPEECH WINDOW SIZE IS ");
  INTEGER_IO.PUT (ITEM => NUMBER, WIDTH => 4);
  NEW_LINE;
      PUT (ITEM => "  4 - NETWORK LOCATION IS ");
      PUT (ITEM => DATABASE);
      NEW_LINE;
      PUT (ITEM => "  5 - WORDS PREFIX IS ");
      PUT (ITEM => PREFIX);
      NEW_LINE;
      PUT (ITEM => "  6 - THE EXTRACTION MODE IS ");
      PUT (ITEM => EXTRACT_MODE);
  NEW_LINE;
  PUT (ITEM => "  7 - THE EXTRACTION COEFFICIENT IS ");
      FLOAT_IO.PUT (ITEM => COEFFICIENT,
             FORE => 2,
             AFT => 2,
             EXP => 0);
  NEW_LINE;
  PUT (ITEM => "  8 - THE MOVING AVERAGE FILTER SIZE IS ");
      INTEGER_IO.PUT (ITEM => FILTER, WIDTH => 2);
  NEW_LINE;
      PUT_LINE (ITEM => "  9 - TO QUIT REVIEW");
  NEW_LINE (2);
      PUT (ITEM => "  NUMBER OF SETTING TO CHANGE -> ");
      CHOICE_INTEGER_IO.GET (ITEM => CHOICE);
      GET_LINE (ITEM => DUMMY);
      case CHOICE is
        when 1 =>
          PUT (ITEM => "NEW VALUE: ");
  INTEGER_IO.GET(START_COUNT);
  GET_LINE(DUMMY);
        when 2 =>
          PUT (ITEM => "NEW VALUE: ");
  INTEGER_IO.GET(END_COUNT);
  GET_LINE(DUMMY);
        when 3 =>
          PUT (ITEM => "NEW VALUE: ");
  INTEGER_IO.GET(NUMBER);
  GET_LINE(DUMMY);
        when 4 =>
          PUT_LINE (ITEM => "NEW VALUE: ");
  GET_LINE(DATABASE);
        when 5 =>
          PUT_LINE (ITEM => "NEW VALUE: ");
  GET_LINE(PREFIX);
        when 6 =>
          PUT_LINE (ITEM => "NEW VALUE: ");
          GET (ITEM => EXTRACT_MODE);
  GET_LINE(DUMMY);
        when 7 =>
          PUT (ITEM => "NEW VALUE: ");
  FLOAT_IO.GET(COEFFICIENT);
  GET_LINE(DUMMY);
        when 8 =>
          PUT (ITEM => "NEW VALUE: ");
  INTEGER_IO.GET(FILTER);
  GET_LINE(DUMMY);
        when 9 =>
          exit REVIEW_LOOP;
      end case;
    exception
      when DATA_ERROR =>
        PUT_LINE (ITEM => "WATCH IT PAL");
    end;
  end loop REVIEW_LOOP;
end REVIEW_EXTRACTION;
```

```
procedure EXTRACTION_CONTROL_PANEL is
  type CHOICE_INTEGER is range 1..12;
  CHOICE : CHOICE_INTEGER;

  package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
  use CHOICE_INTEGER_IO;
begin
  ASSIGN(DATABASE,"CALVIN::DUA0:[TOMR]");
  MAIN: loop
    begin
  NEW_LINE (2);
  PUT_LINE (ITEM => "    1 - INPUT FILE NAME FOR EXTRACTION");
  PUT_LINE (ITEM => "    2 - SET PREFIX FOR EXTRACTED WORDS");
  PUT_LINE (ITEM => "    3 - SET INPUT CHANNEL LOCAL OR NETWORK (NETWORK)");
  PUT_LINE (ITEM => "    4 - REVIEW CURRENT SETTINGS FOR EXTRACTION");
  PUT_LINE (ITEM => "    5 - CREATE LIST OF SPEECH FILE");
  PUT_LINE (ITEM => "    6 - TRANSLATE FILE INTO STANDARD FORMAT");
  PUT_LINE (ITEM => "    7 - CREATE EXTRACTION FEATURE LIST");
  PUT_LINE (ITEM => "    8 - DISPLAY FEATURE LIST");
  PUT_LINE (ITEM => "    9 - EXTRACT WORDS OUT OF FILE");
  PUT_LINE (ITEM => "   10 - MAUNALLY EXTRACT WORDS OUT OF FILE");
  PUT_LINE (ITEM => "   11 - RUN FULL EXTRACTION PROCESS");
  PUT_LINE (ITEM => "   12 - QUIT EXTRACTION OPERATIONS");
  NEW_LINE;
  PUT ("    CHOICE -> ");
  GET (ITEM => CHOICE);
  GET_LINE (ITEM => DUMMY);
  case CHOICE is
    when 1 =>
  PUT_LINE (ITEM => "TYPE NAME OF INPUT FILE NAME, LESS EXTENSION");
  GET_LINE (FILE_NAME);
    when 2 =>
  PUT_LINE (ITEM => "TYPE NAME OF WORD PREFIX");
  GET_LINE(PREFIX);
    when 3 =>
  PUT (ITEM => "THE CURRENT INPUT CHANNEL IS ");
  PUT (ITEM => INPUT_CHANNEL);
  NEW_LINE;
  PUT (ITEM => "NEW INPUT CHANNEL IS ");
  GET (ITEM => INPUT_CHANNEL);
    when 4 =>
  REVIEW_EXTRACTION;
```

```
      when 5 =>
    MAKE_SPEECH_LIST (FILE_NAME);
      when 6 =>
    TRANSLATE (FILE_NAME,INPUT_CHANNEL);
      when 7 =>
    CREATE_EXTRACTION_FEATURE_LIST;
      when 8 =>
    DISPLAY_FEATURE_LIST;
      when 9 =>
    EXTRACT (EXTRACT_MODE);
      when 10 =>
    MANUAL_EXTRACTION;
      when 11 =>
    TRANSLATE (FILE_NAME,INPUT_CHANNEL);
    CREATE_EXTRACTION_FEATURE_LIST;
    EXTRACT (EXTRACT_MODE);
      when 12 =>
    exit MAIN;
    end case;
      exception
    when DATA_ERROR =>
      NEW_LINE;
      PUT_LINE (ITEM => "PLEASE, PLEASE, PLEASE WATCH WHAT YOUR TYPING,
SHEEEEEECH");
      end;
    end loop MAIN;
  end EXTRACTION_CONTROL_PANEL;
end EXTRACTION_PACKAGE;
```

```
- [source_file_header_comment]
with FLOAT_MATH_LIB, TEXT_IO, LINKED_LIST, DATABASE_PACKAGE, STRING_OPERA-
TIONS;
use FLOAT_MATH_LIB, TEXT_IO, DATABASE_PACKAGE, STRING_OPERATIONS;
generic
  NUMBER : in INTEGER := 512;
  M : INTEGER := 20;
package FEATURE_PACKAGE is
- [package_specification_header_comment]
  type FEATURE_ARRAY_TYPE is array (INTEGER range <>) of FLOAT;
  type FEATURE_MODE_TYPE is (SINGLE, BATCH);

  package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);

  - [procedure_header_comment]
  procedure POWER_SPECTRUM (DATA : in FEATURE_ARRAY_TYPE;
        SPECTRUM : OUT FEATURE_ARRAY_TYPE);

  - [procedure_header_comment]
  procedure CREATE_ALL_FEATURE (the_FILE : in VAR_STRING);

  - [procedure_header_comment]
  procedure DO_FEATURE (PROCESSING_MODE : in FEATURE_MODE_TYPE );

  - [procedure_header_comment]
  procedure FEATURE_CONTROL_PANEL;

end FEATURE_PACKAGE;
```

```
- [source_file_header_comment]
package body FEATURE_PACKAGE is
- [package_body_header_comment]

  PREFIX : VAR_STRING;

  function "-" (LEFT, RIGHT : in FEATURE_ARRAY_TYPE) return FEATURE_ARRAY_TYPE is
  - [function_header_comment]
    RESULTS : FEATURE_ARRAY_TYPE(LEFT'RANGE);
  begin
    for INDEX in LEFT'RANGE loop
      RESULTS(INDEX) := LEFT(INDEX)-RIGHT(INDEX);
    end loop;
    return RESULTS;
  end "-";
  function SUMMER (DATA : in FEATURE_ARRAY_TYPE;
       FROM, TO : in INTEGER) return FLOAT is
  -
    SUM : FLOAT := 0.0;
  begin
    for INDEX in FROM..TO loop
      SUM := SUM + DATA(INDEX);
    end loop;
    return SUM;
  end SUMMER;

  function MEL_256 (INTERMEDIATE : in FEATURE_ARRAY_TYPE) return FEA-
TURE_ARRAY_TYPE is
  - [function_header_comment]
    SPECTRUM: FEATURE_ARRAY_TYPE(1..16);
  begin
    SPECTRUM(1) := 10.0*LOG10(SUMMER(INTERMEDIATE,1,4));
    SPECTRUM(2) := 10.0*LOG10(SUMMER(INTERMEDIATE,5,10));
    SPECTRUM(3) := 10.0*LOG10(SUMMER(INTERMEDIATE,11,16));
    SPECTRUM(4) := 10.0*LOG10(SUMMER(INTERMEDIATE,17,23));
    SPECTRUM(5) := 10.0*LOG10(SUMMER(INTERMEDIATE,24,31));
    SPECTRUM(6) := 10.0*LOG10(SUMMER(INTERMEDIATE,32,40));
    SPECTRUM(7) := 10.0*LOG10(SUMMER(INTERMEDIATE,41,51));
    SPECTRUM(8) := 10.0*LOG10(SUMMER(INTERMEDIATE,52,63));
    SPECTRUM(9) := 10.0*LOG10(SUMMER(INTERMEDIATE,64,78));
    SPECTRUM(10) := 10.0*LOG10(SUMMER(INTERMEDIATE,79,94));
    SPECTRUM(11) := 10.0*LOG10(SUMMER(INTERMEDIATE,95,112));
    SPECTRUM(12) := 10.0*LOG10(SUMMER(INTERMEDIATE,113,134));
    SPECTRUM(13) := 10.0*LOG10(SUMMER(INTERMEDIATE,135,158));
    SPECTRUM(14) := 10.0*LOG10(SUMMER(INTERMEDIATE,159,186));
    SPECTRUM(15) := 10.0*LOG10(SUMMER(INTERMEDIATE,187,219));
    SPECTRUM(16) := 10.0*LOG10(SUMMER(INTERMEDIATE,220,256));
    return SPECTRUM;
  end MEL_256;

  function MEL_128 (INTERMEDIATE : in FEATURE_ARRAY_TYPE) return FEA-
TURE_ARRAY_TYPE is
  - [function_header_comment]
     SPECTRUM: FEATURE_ARRAY_TYPE(1..16);
  begin
    SPECTRUM(1) := 10.0*LOG10(SUMMER(INTERMEDIATE,1,2));
    SPECTRUM(2) := 10.0*LOG10(SUMMER(INTERMEDIATE,3,5));
    SPECTRUM(3) := 10.0*LOG10(SUMMER(INTERMEDIATE,6,8));
    SPECTRUM(4) := 10.0*LOG10(SUMMER(INTERMEDIATE,9,11));
    SPECTRUM(5) := 10.0*LOG10(SUMMER(INTERMEDIATE,12,15));
    SPECTRUM(6) := 10.0*LOG10(SUMMER(INTERMEDIATE,16,20));
    SPECTRUM(7) := 10.0*LOG10(SUMMER(INTERMEDIATE,21,25));
    SPECTRUM(8) := 10.0*LOG10(SUMMER(INTERMEDIATE,26,32));
    SPECTRUM(9) := 10.0*LOG10(SUMMER(INTERMEDIATE,33,39));
    SPECTRUM(10) := 10.0*LOG10(SUMMER(INTERMEDIATE,40,47));
    SPECTRUM(11) := 10.0*LOG10(SUMMER(INTERMEDIATE,48,56));
    SPECTRUM(12) := 10.0*LOG10(SUMMER(INTERMEDIATE,57,67));
    SPECTRUM(13) := 10.0*LOG10(SUMMER(INTERMEDIATE,68,79));
    SPECTRUM(14) := 10.0*LOG10(SUMMER(INTERMEDIATE,88,93));
    SPECTRUM(15) := 10.0*LOG10(SUMMER(INTERMEDIATE,94,109));
    SPECTRUM(16) := 10.0*LOG10(SUMMER(INTERMEDIATE,110,128));

     return SPECTRUM;
  end MEL_128;

  function MEL_DIGITAL_FILTER (N : in INTEGER;
     LINEAR_FREQS : in FEATURE_ARRAY_TYPE;
     ALPHA : in FLOAT ) return FEATURE_ARRAY_TYPE is
  - [function_header_comment]
     TEMP1, TEMP, NEW_FREQS : FEATURE_ARRAY_TYPE(LINEAR_FREQS'RANGE);
     NEW_A, SMALL : FLOAT := 0.1E-20;
-    N : INTEGER := LINEAR_FREQS'LAST;
  begin
    TEMP(N) := LINEAR_FREQS(N);
```

```
for I in reverse 0..N-1 loop
  TEMP(I) := LINEAR_FREQS(I) + (ALPHA * TEMP(I+1));
  if ABS(TEMP(I)) < SMALL then
    TEMP(I) := 0.0;
  end if;
end loop;
NEW_FREQS(0) := LINEAR_FREQS(0) + (ALPHA * TEMP(1));
TEMP1(N) := 0.0;
for I in reverse 0..N-1 loop
  TEMP1(I) := TEMP(I+1) + (ALPHA * TEMP1(I+1));
  if ABS(TEMP1(I)) < SMALL then
    TEMP(I) := 0.0;
  end if;
end loop;

NEW_FREQS(1) := TEMP(1) + (ALPHA * TEMP1(1));
for K in 2..N loop
  if K MOD 2 = 1 then
    TEMP1(N) := 0.0;
    for I in reverse 0..N-1 loop
      TEMP1(I) := TEMP(I+1) + (ALPHA * (TEMP1(I+1) - TEMP(I)));
      if ABS(TEMP1(I)) < SMALL then
        TEMP1(I) := 0.0;
      end if;
    end loop;
NEW_FREQS(K) := TEMP(1) + (ALPHA * (TEMP1(1) - NEW_FREQS(K-1)));
  else
    TEMP(N) := 0.0;
    for I in reverse 0..N-1 loop
      TEMP(I) := TEMP1(I+1) + (ALPHA * (TEMP(I+1) - TEMP1(I)));
      if ABS(TEMP(I)) < SMALL then
        TEMP(I) := 0.0;
      end if;
    end loop;
NEW_FREQS(K) := TEMP1(1) + (ALPHA * (TEMP(1) - NEW_FREQS(K-1)));
  end if;
end loop;
NEW_A := 1.0 - ALPHA**2;
for I in 1..N loop
  NEW_FREQS(I) := NEW_FREQS(I) * NEW_A;
end loop;
return NEW_FREQS;
```

```
  end MEL_DIGITAL_FILTER;


  function MEL_FILTER (FREQUENCY_ARRAY : in FEATURE_ARRAY_TYPE) return FEA-
TURE_ARRAY_TYPE is
  - [function_header_comment]
     FILTERS : FEATURE_ARRAY_TYPE(1..M);
    WEIGHT, THIS_FREQ, LOW_FREQ, MID_FREQ, HI_FREQ, SUM, BIN_SIZE, MEL_MAX,
MEL_BIN : FLOAT;


    function MEL_SCALE (MEL_BIN : in FLOAT ) return FLOAT is
    - [function_header_comment]
      BIN : FLOAT;
    begin
      BIN := (10.0**(MEL_BIN*LOG10(2.0)/1000.0)-1.0)*1000.0;
      return BIN;
    end MEL_SCALE;


  begin
    BIN_SIZE := 8000.0/FLOAT(NUMBER/2);
    MEL_MAX := 1000.0/LOG(2.0)*LOG(9.0);
    MEL_BIN := MEL_MAX/FLOAT(M);
    for INDEX in 1..M loop
      LOW_FREQ := MEL_SCALE(MEL_BIN*FLOAT(INDEX-1));
      MID_FREQ := MEL_SCALE(MEL_BIN*FLOAT(INDEX));
      HI_FREQ := MEL_SCALE(MEL_BIN*FLOAT(INDEX+1));
      SUM := 0.0;
      for INDEX1 in !..NUMBER/2 loop
    THIS_FREQ := BIN_SIZE*FLOAT(INDEX1);
        if THIS_FREQ > LOW_FREQ AND THIS_FREQ < HI_FREQ then
          if THIS_FREQ < MID_FREQ then
            WEIGHT := (THIS_FREQ-LOW_FREQ)/(MID_FREQ-LOW_FREQ);
          else
            WEIGHT := 1.0 - (THIS_FREQ-MID_FREQ)/(HI_FREQ-MID_FREQ);
          end if;
          SUM := SUM + WEIGHT*FREQUENCY_ARRAY(INDEX1);
        end if;
      end loop;
      FILTERS(INDEX) := LOG10(SUM);
    end loop;
    return FILTERS·
  end MEL_FILTER;
```

```
function DBPEROCTIVE (VALUE : in FLOAT; INDEX : in INTEGER) return FLOAT is
- [function_header_comment]
  DUMMY, RESULT : FLOAT;
begin
  DUMMY := 8000.0/(FLOAT(NUMBER)/2.0)*FLOAT(INDEX);
  if DUMMY > 600.0 then
    RESULT := VALUE * DUMMY/600.0;
  else
    RESULT := VALUE;
  end if;
  return RESULT;
end DBPEROCTIVE;

procedure LP_COEFFICIENTS (DATA : in FEATURE_ARRAY_TYPE;
    PM : in out FLOAT;
    LP_FILTER_COEFFICIENTS : in out  FEATURE_ARRAY_TYPE) is
- [function_header_comment]
  WK1, WK2 : FEATURE_ARRAY_TYPE(1..NUMBER);
  RESULTS, WKM : FEATURE_ARRAY_TYPE(1..M);
  P,PNEUM, DENOM : FLOAT;
begin
  P := 0.0;
  for j in 1..NUMBER loop
    P := P+DATA(J)**2;
  end loop;
  PM := P/FLOAT(NUMBER);
  WK1(1) := DATA(1);
  WK2(NUMBER-1) := DATA(NUMBER);
  for J in 2..NUMBER-1 loop
    WK1(J) := DATA(J);
    WK2(J-1) := DATA(J);
  end loop;
  for K in 1..M loop
    PNEUM := 0.0;
    DENOM := 0.0;
    for J in 1..NUMBER-K loop
  PNEUM := PNEUM+WK1(J)*WK2(J);
  DENOM := DENOM+WK1(J)**2+WK2(J)**2;
    end loop;
    LP_FILTER_COEFFICIENTS(K) := 2.0*PNEUM/DENOM;
        PM := PM*(1.0-LP_FILTER_COEFFICIENTS(K)**2);

    if K /= 1 then
  for I in 1..K-1 loop
    LP_FILTER_COEFFICIENTS(I) := WKM(I)-LP_FILTER_COEFFICIENTS(K)*WKM(K-I);
  end loop;
    end if;
    if K /= M then
  for I in 1..K loop
    WKM(I) := LP_FILTER_COEFFICIENTS(I);
  end loop;
  for J in 1..NUMBER-K-1 loop
    WK1(J) := WK1(J)-WKM(K)*WK2(J);
    WK2(J) := WK2(J+1)-WKM(K)*WK1(J+1);
  end loop;
    end if;
  end loop;
end LP_COEFFICIENTS;

function LPC_POWER_SPECTRUM (LP_FILTER_COEFFICIENTS : in FEATURE_ARRAY_TYPE;
   FDT, PM : in FLOAT) return FLOAT is
- [function_header_comment]
  WR, WI, WTEMP, SUMR, SUMI : FLOAT;
  THETA : FLOAT := 6.28318530717959*FDT;
  WPR : FLOAT := COS(THETA);
  WPI : FLOAT := SIN(THETA);
begin
  WR := 1.0;
  WI := 0.0;
  SUMR := 1.0;
  SUMI := 0.0;
  for I in 1..M loop
    WTEMP := WR;
    WR := WR*WPR-WI*WPI;
    WI := WI*WPR+WTEMP*WPI;
    SUMR := SUMR-LP_FILTER_COEFFICIENTS(I)*WR;
    SUMI := SUMI-LP_FILTER_COEFFICIENTS(I)*WI;
  end loop;
  return PM/(SUMR**2+SUMI**2);
end LPC_POWER_SPECTRUM;

procedure POWER_SPECTRUM (DATA : in FEATURE_ARRAY_TYPE;
      SPECTRUM : OUT FEATURE_ARRAY_TYPE) is
- [function_header_comment]
```

```
    LP_FILTER_COEFFICIENTS : FEATURE_ARRAY_TYPE(1..NUMBER);
    LPC_INTERMEDIATE : FEATURE_ARRAY_TYPE(1..NUMBER/2);
    LOG_ENERGY : FEATURE_ARRAY_TYPE(1..M);
    GAIN, SUM, DUMMY, N2, FDT : FLOAT;

  begin

    LP_COEFFICIENTS(DATA,GAIN,LP_FILTER_COEFFICIENTS);

    N2 := FLOAT(NUMBER/2)*2.0;
    for INDEX in 0..NUMBER/2-1 loop
      DUMMY := LPC_POWER_SPECTRUM (LP_FILTER_COEFFI-
CIENTS,FLOAT(INDEX)/N2,GAIN);
      LPC_INTERMEDIATE(INDEX+1) := DBPEROCTIVE(DUMMY,INDEX+1);
    end loop;

- GENERATE MELED SPECTRUM

    if NUMBER = 512 then
      SPECTRUM(1..16) := MEL_256(LPC_INTERMEDIATE);
    elsif NUMBER = 256 then
      SPECTRUM(1..16) := MEL_128(LPC_INTERMEDIATE);
    end if;

  end POWER_SPECTRUM;

  procedure LP_SPECTRUM (LP_FILTER_COEFFICIENTS : in FEATURE_ARRAY_TYPE;
      SPECTRUM, CEPSTRUM, MEL_CEPSTRUM : OUT FEATURE_ARRAY_TYPE;
      LP_CEPSTRUM : in out FEATURE_ARRAY_TYPE;
      GAIN : in FLOAT) is
  - [function_header_comment]

    LPC_INTERMEDIATE : FEATURE_ARRAY_TYPE(1..NUMBER/2);
    LOG_ENERGY : FEATURE_ARRAY_TYPE(1..M);
    SUM, DUMMY, N2, FDT : FLOAT;

  begin
    N2 := FLOAT(NUMBER/2)*2.0;
    for INDEX in 0..NUMBER/2-1 loop
      DUMMY := LPC_POWER_SPECTRUM (LP_FILTER_COEFFI-
CIENTS,FLOAT(INDEX)/N2,GAIN);
      LPC_INTERMEDIATE(INDEX+1) := DBPEROCTIVE(DUMMY,INDEX+1);
    end loop;
```

```
- GENERATE MELED SPECTRUM

    SPECTRUM(1..16) := MEL_256(LPC_INTERMEDIATE);

- GENERATE LF CEPSTRUM COEEFIICENTS

    for INDEX in 1..M loop
      SUM := 0.0;
      for K in 1..NUMBER/2 loop
        SUM := SUM + LOG10(LPC_INTERMEDI-
ATE(K))*COS(3.1415*FLOAT(INDEX*K)/FLOAT(NUMBER));
      end loop;
      CEPSTRUM(INDEX) := SUM;
    end loop;

- GENERATE LP CEPSTRUM COEEFIICENTS

    for INDEX in 1..M loop
      SUM := 0.0;
      for K in 1..INDEX-1 loop
        SUM := SUM + LP_FILTER_COEFFICIENTS(K)*LP_CEPSTRUM(INDEX-K)*FLOAT((K-
INDEX)/INDEX);
      end loop;
      LP_CEPSTRUM(INDEX) := LP_FILTER_COEFFICIENTS(INDEX)+ SUM;
    end loop;

- GENERATE MEL CEPSTRUM COEFFICIENTS

    LOG_ENERGY := MEL_FILTER(LPC_INTERMEDIATE);
    for INDEX in 1..M loop
      SUM  := 0.0;
      for K in 1..M loop
        SUM := SUM + LOG_ENERGY(K)*COS(FLOAT(INDEX)*(FLOAT(K)-
0.5)*3.1415/FLOAT(M));
      end loop;
      MEL_CEPSTRUM(INDEX) := SUM;
    end loop;

  end LP_SPECTRUM;

  procedure FILE_ONLINE (FILE : in VAR_STRING) is
  - [procedure_header_comment]
```

```
    DATUM : FLOAT;
    FILE_NAME : VAR_STRING;
    INPUT_FILE : FILE_TYPE;

    package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
     use FLOAT_IO;
  begin

    SPEECH_LIST.CLEAR_LIST(SPEECH_HEAD,SPEECH_TAIL);
    FILE_NAME := DIRECTORY_NAME&FILE&".DIG";
    OPEN (
      FILE => INPUT_FILE,
      MODE => IN_FILE,
      NAME => FILE_NAME);
    loop
      GET (FILE => INPUT_FILE, ITEM => DATUM);
      SPEECH_LIST.ADD_TO(DATUM,SPEECH_HEAD,SPEECH_TAIL);
    end loop;
    exception
          when END_ERROR =>

    CLOSE (FILE => INPUT_FILE);
  end FILE_ONLINE;

  procedure CREATE_ALL_FEATURE (THE_FILE : in VAR_STRING) is
  - [procedure_header_comment]

     type FEATURE_TYPES is array (0..M+1) of FLOAT;
     type OUTPUT_FILES_TYPE is array (1..8) of FILE_TYPE;
     type NAMES is array (1..8) of VAR_STRING;

    LP_FILTER_COEFFICIENTS : FEATURE_ARRAY_TYPE(0..M+1);
    DELTA_BL_CEPSTRUM, BL_CEPSTRUM, DELTA_MEL_CEPSTRUM, DELTA_SPECTRUM,
SPECTRUM : FEATURE_ARRAY_TYPE(0..M+1);
    DELTA_CEPSTRUM, LP_CEPSTRUM, MEL_CEPSTRUM, CEPSTRUM : FEA-
TURE_ARRAY_TYPE(0..M+1);
    TWO_BACK1, TWO_BACK2, TWO_BACK3, TWO_BACK4 : FEA-
TURE_ARRAY_TYPE(0..M+1);
    ONE_BACK1, ONE_BACK2, ONE_BACK3, ONE_BACK4 : FEATURE_ARRAY_TYPE(0..M+1);
    REFLECTION : FEATURE_ARRAY_TYPE(0..M+1) := (OTHERS => 0.0);
    NORMALIZE : FEATURE_ARRAY_TYPE(1..8) := (OTHERS => 0.0);
    RAW_SPEECH : FEATURE_ARRAY_TYPE(1..NUMBER);
    OUTPUT_FILES : OUTPUT_FILES_TYPE;
    GAIN, WARP : FLOAT;
    FILE_NAMES : NAMES;
    M_COMPONENTS,COUNTER : INTEGER := 0;
    PASS : INTEGER := 1;
    SPEECH_POINTER : SPEECH_LIST.LIST;

    package FEATURE_LIST is new LINKED_LIST (ITEM_TYPE => FEATURE_TYPES);
    package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
    package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
    use FLOAT_IO, INTEGER_IO, FEATURE_LIST;

    FEATURE_HEAD, FEATURE_TAIL : FEATURE_LIST.LIST;

    function NORMAL (FROM, TO : in INTEGER;
     FEATURE_VARIABLE : in FEATURE_ARRAY_TYPE) return FLOAT is
    - [function_header_comment]
      NORM : FLOAT := 0.0;
    begin
      for INDEX in FROM..TO loop
        NORM := NORM + FEATURE_VARIABLE(INDEX)**2;
      end loop;
      return NORM;
    end NORMAL;

  begin
    FILE_ONLINE(THE_FILE);
    SPEECH_POINTER := SPEECH_HEAD;
    CLEAR_LIST(FEATURE_HEAD,FEATURE_TAIL);
    M_COMPONENTS := 16;
    FILE_NAMES(1) := DIRECTORY_NAME&THE_FILE&".LPPS";
    FILE_NAMES(2) := DIRECTORY_NAME&THE_FILE&".DPPS";
    FILE_NAMES(3) := DIRECTORY_NAME&THE_FILE&".LFCC";
    FILE_NAMES(4) := DIRECTORY_NAME&THE_FILE&".DFCC";
    FILE_NAMES(5) := DIRECTORY_NAME&THE_FILE&".MFMC";
    FILE_NAMES(6) := DIRECTORY_NAME&THE_FILE&".DFMC";
    FILE_NAMES(7) := DIRECTORY_NAME&THE_FILE&".BLMC";
    FILE_NAMES(8) := DIRECTORY_NAME&THE_FILE&".DLMC";
    for INDEX in 1..8 loop
      CREATE (
        FILE => OUTPUT_FILES(INDEX),
        MODE => OUT_FILE,
        NAME => FILE_NAMES(INDEX));
  end loop;

  loop
    for INDEX in 1..NUMBER loop
  COUNTER := INDEX;
  RAW_SPEECH(INDEX) := SPEECH_LIST.GET_NODE(SPEECH_POINTER);
  SPEECH_POINTER := SPEECH_LIST.NEXT_LINK(SPEECH_POINTER);
    end loop;
    LP_COEFFICIENTS(RAW_SPEECH,GAIN,LP_FILTER_COEFFICIENTS);
    LP_SPECTRUM(LP_FILTER_COEFFICIENTS,SPECTRUM,CEPSTRUM,MEL_CEP-
STRUM,LP_CEPSTRUM,GAIN);
    BL_CEPSTRUM(0..M-1) := CEPSTRUM(1..M);
    BL_CEPSTRUM := MEL_DIGITAL_FILTER(M-1,BL_CEPSTRUM,0.6);
    BL_CEPSTRUM(1..M) := BL_CEPSTRUM(0..M-1);
    NORMALIZE(1) := NORMALIZE(1) + NORMAL(1,16,SPECTRUM);
     ADD_TO(FEATURE_TYPES(SPECTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(3) := NORMALIZE(3) + NORMAL(1,M_COMPONENTS,CEPSTRUM);
     ADD_TO(FEATURE_TYPES(CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(5) := NORMALIZE(5) + NORMAL(1,M_COMPONENTS,MEL_CEPSTRUM);
     ADD_TO(FEATURE_TYPES(MEL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(7) := NORMALIZE(7) + NORMAL(1,M_COMPONENTS,BL_CEPSTRUM);
     ADD_TO(FEATURE_TYPES(BL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
     if PASS = 1 then
  PASS := PASS + 1;
       TWO_BACK1 := MEL_CEPSTRUM;
  TWO_BACK2 := SPECTRUM;
  TWO_BACK3 := CEPSTRUM;
  TWO_BACK4 := BL_CEPSTRUM;
     elsif PASS = 2  then
       PASS := PASS + 1;
       ONE_BACK1 := MEL_CEPSTRUM;
  ONE_BACK2 := SPECTRUM;
  ONE_BACK3 := CEPSTRUM;
  ONE_BACK4 := BL_CEPSTRUM;
  ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
  ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
  ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
  ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
     else
  PASS := PASS + 1;
  DELTA_MEL_CEPSTRUM := MEL_CEPSTRUM-TWO_BACK1;
  TWO_BACK1 := ONE_BACK1;
```

```
      ONE_BACK1 := MEL_CEPSTRUM;
      DELTA_SPECTRUM := SPECTRUM-TWO_BACK2;
      TWO_BACK2 := ONE_BACK2;
      ONE_BACK2 := SPECTRUM;
      DELTA_CEPSTRUM := CEPSTRUM-TWO_BACK3;
      TWO_BACK3 := ONE_BACK3;
      ONE_BACK3 := CEPSTRUM;
      DELTA_BL_CEPSTRUM := BL_CEPSTRUM-TWO_BACK4;
      TWO_BACK4 := ONE_BACK4;
      ONE_BACK4 := BL_CEPSTRUM;
      NORMALIZE(2) := NORMALIZE(2) + NORMAL(1,16,DELTA_SPECTRUM);
      ADD_TO(FEATURE_TYPES(DELTA_SPECTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(4) := NORMALIZE(4) + NORMAL(1,M_COMPONENTS,DELTA_CEPSTRUM);
      ADD_TO(FEATURE_TYPES(DELTA_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(6) := NORMALIZE(6) + NORMAL(1,M_COMPONENTS,DELTA_MEL_CEP-
  STRUM);
      ADD_TO(FEATURE_TYPES(DELTA_MEL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(8) := NORMALIZE(8) + NORMAL(1,M_COMPONENTS,DELTA_BL_CEP-
  STRUM);
      ADD_TO(FEATURE_TYPES(DELTA_BL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
        end if;
      end loop;
    exception
      when SPEECH_LIST.UNDER_FLOW =>
        if COUNTER /= 1 then
      for INDEX in COUNTER..NUMBER loop
        RAW_SPEECH(INDEX) := 0.0;
      end loop;
      LP_COEFFICIENTS(RAW_SPEECH,GAIN,LP_FILTER_COEFFICIENTS);
      LP_SPECTRUM(LP_FILTER_COEFFICIENTS,SPECTRUM,CEPSTRUM,MEL_CEP-
  STRUM,LP_CEPSTRUM,GAIN);
      BL_CEPSTRUM(0..M-1) := CEPSTRUM(1..M);
      BL_CEPSTRUM := MEL_DIGITAL_FILTER(M-1,BL_CEPSTRUM,0.6);
      BL_CEPSTRUM(1..M) := BL_CEPSTRUM(0..M-1);
      NORMALIZE(1) := NORMALIZE(1) + NORMAL(1,16,SPECTRUM);
      ADD_TO(FEATURE_TYPES(SPECTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(3) := NORMALIZE(3) + NORMAL(1,M_COMPONENTS,CEPSTRUM);
      ADD_TO(FEATURE_TYPES(CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(5) := NORMALIZE(5) + NORMAL(1,M_COMPONENTS,MEL_CEPSTRUM);
      ADD_TO(FEATURE_TYPES(MEL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      NORMALIZE(7) := NORMALIZE(7) + NORMAL(1,M_COMPONENTS,BL_CEPSTRUM);
      ADD_TO(FEATURE_TYPES(BL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      DELTA_MEL_CEPSTRUM := MEL_CEPSTRUM-TWO_BACK1;
    DELTA_SPECTRUM := SPECTRUM-TWO_BACK2;
    DELTA_CEPSTRUM := CEPSTRUM-TWO_BACK3;
    DELTA_BL_CEPSTRUM := BL_CEPSTRUM-TWO_BACK4;
    NORMALIZE(2) := NORMALIZE(2) + NORMAL(1,16,DELTA_SPECTRUM);
    ADD_TO(FEATURE_TYPES(DELTA_SPECTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(4) := NORMALIZE(4) + NORMAL(1,M_COMPONENTS,DELTA_CEPSTRUM);
    ADD_TO(FEATURE_TYPES(DELTA_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(6) := NORMALIZE(6) + NORMAL(1,M_COMPONENTS,DELTA_MEL_CEP-
STRUM);
    ADD_TO(FEATURE_TYPES(DELTA_MEL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
    NORMALIZE(8) := NORMALIZE(8) + NORMAL(1,M_COMPONENTS,DELTA_BL_CEP-
STRUM);
    ADD_TO(FEATURE_TYPES(DELTA_BL_CEPSTRUM),FEATURE_HEAD,FEATURE_TAIL);
      else
        PASS := PASS - 1;
      end if;
       ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
       ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
       ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
       ADD_TO(FEATURE_TYPES(REFLECTION),FEATURE_HEAD,FEATURE_TAIL);
      for INDEX in 1..8 loop
        NORMALIZE(INDEX) := SQRT(NORMALIZE(INDEX));
      end loop;
      for INDEX in 1..PASS loop
    if INDEX = 1 then
       REMOVE_NEXT_NODE(FEATURE_TYPES(SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..16 loop
    PUT (FILE => OUTPUT_FILES(1), ITEM => SPECTRUM(INDEX)/NORMALIZE(1),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(1), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(3), ITEM => CEPSTRUM(INDEX)/NORMALIZE(3),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(3), SPACING => 1);
      end loop;
```

```
        REMOVE_NEXT_NODE(FEATURE_TYPES(MEL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
     PUT (FILE => OUTPUT_FILES(5), ITEM => MEL_CEPSTRUM(INDEX)/NORMALIZE(5),
     FORE => 2,
        AFT => 8,
        EXP => 0);
     NEW_LINE (FILE => OUTPUT_FILES(5), SPACING => 1);
        end loop;
        REMOVE_NEXT_NODE(FEATURE_TYPES(BL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
     PUT (FILE => OUTPUT_FILES(7), ITEM => BL_CEPSTRUM(INDEX)/NORMALIZE(7),
        FORE => 2,
        AFT => 8,
        EXP => 0);
     NEW_LINE (FILE => OUTPUT_FILES(7), SPACING => 1);
        end loop;
     elsif INDEX in 2..PASS-1 then
        REMOVE_NEXT_NODE(FEATURE_TYPES(SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..16 loop
     PUT (FILE => OUTPUT_FILES(1), ITEM => SPECTRUM(INDEX)/NORMALIZE(1),
        FORE => 2,
        AFT => 8,
        EXP => 0);
     NEW_LINE (FILE => OUTPUT_FILES(1), SPACING => 1);
        end loop;
        REMOVE_NEXT_NODE(FEATURE_TYPES(CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
     PUT (FILE => OUTPUT_FILES(3), ITEM => CEPSTRUM(INDEX)/NORMALIZE(3),
        FORE => 2,
        AFT => 8,
        EXP => 0);
     NEW_LINE (FILE => OUTPUT_FILES(3), SPACING => 1);
        end loop;
        REMOVE_NEXT_NODE(FEATURE_TYPES(MEL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
     PUT (FILE => OUTPUT_FILES(5), ITEM => MEL_CEPSTRUM(INDEX)/NORMALIZE(5),
        FORE => 2,
        AFT => 8,
        EXP => 0);
```

```
    NEW_LINE (FILE => OUTPUT_FILES(5), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(BL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(7), ITEM => BL_CEPSTRUM(INDEX)/NORMALIZE(7),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(7), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..16 loop
    PUT (FILE => OUTPUT_FILES(2), ITEM => DELTA_SPECTRUM(INDEX)/NORMALIZE(2),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(2), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(4), ITEM => DELTA_CEPSTRUM(INDEX)/NORMALIZE(4),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(4), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_MEL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(6), ITEM => DELTA_MEL_CEPSTRUM(INDEX)/NORMAL-
IZE(6),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(6), SPACING => 1);
    end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_BL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(8), ITEM => DELTA_BL_CEPSTRUM(INDEX)/NORMALIZE(8),
      FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(8), SPACING => 1);
        end loop;
      else
         REMOVE_NEXT_NODE(FEATURE_TYPES(SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..16 loop
      PUT (FILE => OUTPUT_FILES(1), ITEM => SPECTRUM(INDEX)/NORMALIZE(1),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(1), SPACING => 1);
        end loop;
         REMOVE_NEXT_NODE(FEATURE_TYPES(CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(3), ITEM => CEPSTRUM(INDEX)/NORMALIZE(3),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(3), SPACING => 1);
        end loop;
         REMOVE_NEXT_NODE(FEATURE_TYPES(MEL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(5), ITEM => MEL_CEPSTRUM(INDEX)/NORMALIZE(5),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(5), SPACING => 1);
        end loop;
         REMOVE_NEXT_NODE(FEATURE_TYPES(BL_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(7), ITEM => BL_CEPSTRUM(INDEX)/NORMALIZE(7),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(7), SPACING => 1);
        end loop;
         REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..16 loop
```

```
    PUT (FILE => OUTPUT_FILES(2), ITEM => DELTA_SPECTRUM(INDEX)/NORMALIZE(2),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(2), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(4), ITEM => DELTA_CEPSTRUM(INDEX)/NORMALIZE(4),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(4), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_MEL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(6), ITEM => DELTA_MEL_CEPSTRUM(INDEX)/NORMAL-
IZE(6),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(6), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_BL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
      for INDEX in 1..M_COMPONENTS loop
    PUT (FILE => OUTPUT_FILES(8), ITEM => DELTA_BL_CEPSTRUM(INDEX)/NORMALIZE(8),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(8), SPACING => 1);
      end loop;
       REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_SPECTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
      for INDEX in 1..16 loop
    PUT (FILE => OUTPUT_FILES(2), ITEM => DELTA_SPECTRUM(INDEX)/NORMALIZE(2),
      FORE => 2,
      AFT => 8,
      EXP => 0);
    NEW_LINE (FILE => OUTPUT_FILES(2). SPACING => 1);
      end loop;
```

```
        REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_CEPSTRUM),FEATURE_HEAD,FEA-
TURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(4), ITEM => DELTA_CEPSTRUM(INDEX)/NORMALIZE(4),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(4), SPACING => 1);
        end loop;
        REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_MEL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(6), ITEM => DELTA_MEL_CEPSTRUM(INDEX)/NORMAL-
IZE(6),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(6), SPACING => 1);
        end loop;
        REMOVE_NEXT_NODE(FEATURE_TYPES(DELTA_BL_CEPSTRUM),FEA-
TURE_HEAD,FEATURE_TAIL);
        for INDEX in 1..M_COMPONENTS loop
      PUT (FILE => OUTPUT_FILES(8), ITEM => DELTA_BL_CEPSTRUM(INDEX)/NORMALIZE(8),
        FORE => 2,
        AFT => 8,
        EXP => 0);
      NEW_LINE (FILE => OUTPUT_FILES(8), SPACING => 1);
        end loop;
      end if;
        end loop;
        for INDEX in 1..8 loop
      CLOSE (FILE => OUTPUT_FILES(INDEX));
        end loop;
    end CREATE_ALL_FEATURE;

    procedure DO_FEATURE (PROCESSING_MODE : in FEATURE_MODE_TYPE ) is
    - [procedure_header_comment]
      OUTPUT_FILE : FILE_TYPE;
      WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
      THE_FILE, FILE_NAME : VAR_STRING;
    begin
      if PROCESSING_MODE = SINGLE then
        PUT_LINE (ITEM => "TYPE IN NAME OF WORD, LESS EXTENSION");
    GET_LINE (ITEM => THE_FILE);
  else
    NEXT_WORD(THE_FILE,WORD_POINTER);
  end if;
  FEATURES: loop
    THE_FILE := PREFIX&THE_FILE;
    PUT (ITEM => "CREATING FEATURE(S) FOR WORD ");
    PUT (ITEM => THE_FILE);
    NEW_LINE;
    CREATE_ALL_FEATURE(THE_FILE);
    if PROCESSING_MODE = SINGLE then
  exit FEATURES;
    else
  NEXT_WORD(THE_FILE,WORD_POINTER);
    end if;
  end loop FEATURES;
exception
  when DATABASE_LIST.UNDER_FLOW =>
    NULL;
end DO_FEATURE;

procedure FEATURE_CONTROL_PANEL is
- [procedure_header_comment]

  subtype CHOICE_INTEGER is INTEGER range 1..4;
  INPUT_FILE, OUTPUT_FILE : FILE_TYPE;
  THE_FILE, DUMMY, UNDER_SCORE, PERSON, EXTENSION, FILE_NAME : VAR_STRING;
  CHOICE : CHOICE_INTEGER;
  MODE : FEATURE_MODE_TYPE := BATCH;

  package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (FEATURE_MODE_TYPE);
  package INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
  use INTEGER_IO, ENUMERATION_IO;

begin
  MAIN: loop
    begin
  NEW_LINE (2);
  PUT_LINE (ITEM => "     1 - SET MODE - SINGLE OR BATCH (BATCH)");
  PUT_LINE (ITEM => "     2 - SET PREFIX NAME");
  PUT_LINE (ITEM => "     3 - CREATE ALL FEATURES");
  PUT_LINE (ITEM => "     4 - TO QUIT");
  NEW_LINE;
  PUT (ITEM => "     CHOICE -> ");
  GET(CHOICE);
  GET_LINE (ITEM => DUMMY);
  case CHOICE is
    when 1 =>
  NEW_LINE;
  PUT (ITEM => "TYPE SINGLE OR BATCH ");
  GET (ITEM => MODE);
  GET_LINE (ITEM => DUMMY);
    when 2 =>
  NEW_LINE;
  PUT (ITEM => "TYPE IN NAME OF FEATURE ");
  GET_LINE (ITEM => PREFIX);
    when 3 =>
  DO_FEATURE(MODE);
    when 4 =>
  exit MAIN;
  end case;
    exception
  when DATA_ERROR =>
    PUT_LINE (ITEM => "FIRST DAY WITH THE NEW FINGERS SHEEEESH, TRY AGAIN");
    end;
  end loop MAIN;
 end FEATURE_CONTROL_PANEL;
end FEATURE_PACKAGE;
```

```
- [source_file_header_comment]
with TEXT_IO, REC_SUPPORT_PACKAGE, DATABASE_PACKAGE, STRING_OPERATIONS;
with FUSION_REC_PACKAGE;
use TEXT_IO, DATABASE_PACKAGE, STRING_OPERATIONS, FUSION_REC_PACKAGE;
package RECOGNITION_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    This package instantiates the rec_support package for all the different
-  used for recognition
-
- [optional package tags]
- -
  use FUSION_LIST, FUSED_FEATURE_LIST;

  t,pe RECOGNIZER_TYPE is (TIME_WARP, TWO_DEE, ERROR);
  M : CONSTANT := 16;

  package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (RECOGNIZER_TYPE);

  - instantiate packages for each feature
  package LPPS_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => 16,
      NAME_OF_FEATURE => "LPPS"
      );
  package DPPS_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => 16,
      NAME_OF_FEATURE => "DPPS"
      );
  package LFCC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "LFCC"
      );
  package DFCC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "DFCC"
      );
  package MFMC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS =>
M,
      NAME_OF_FEATURE => "MFMC"
      );
  package DFMC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "DFMC"
      );
  package BLMC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
     NAME_OF_FEATURE => "BLMC"
     );
  package DLMC_SPEECH is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
     NAME_OF_FEATURE => "DLMC"
     );
  use LPPS_SPEECH, DPPS_SPEECH, LFCC_SPEECH, DFCC_SPEECH;
  use MFMC_SPEECH, DFMC_SPEECH, BLMC_SPEECH, DLMC_SPEECH, ENUMERATION_IO;

  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    This procedure allows for an interactive menu for doing recognition
  -  operations
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure RECOGNITION_CONTROL_PANEL;

end RECOGNITION_PACKAGE;
```

```
- [source_file_header_comment]
package body RECOGNITION_PACKAGE is
- [package_body_header_comment]
  procedure RECOGNITION_CONTROL_PANEL is
  - [procedure_header_comment]
    subtype CHOICE_INTEGER is INTEGER range 1..15;
    CHOICE : CHOICE_INTEGER;
    ONLINE : CHARACTER;
    RECOGNIZER : RECOGNIZER_TYPE := TIME_WARP;
    COMPONENTS : INTEGER;
    WORD1, WORD2, TEST1, TEST2, THE_FILE, DUMMY, NEW_FEATURE_NAME :
VAR_STRING;
    RESULTS : RESULT_RECORD_TYPE;
    NEW_FEATURES : FEATURE_RECORD_TYPE;
    FUSED_FEATURE_HEAD, FUSED_FEATURE_TAIL : FUSED_FEATURE_LIST.LIST;
    package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
    package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
    use CHOICE_INTEGER_IO, INTEGER_IO;

  begin
    FUSED_FEATURE_LIST.NULL_POINTER(FUSED_FEATURE_HEAD);
    FUSED_FEATURE_LIST.NULL_POINTER(FUSED_FEATURE_TAIL);
    RESULT_LIST.NULL_POINTER(RESULTS.RESULT_HEAD);
    RESULT_LIST.NULL_POINTER(RESULTS.RESULT_TAIL);
    MAIN: loop
      begin
    NEW_LINE(2);
    PUT_LINE (ITEM => "   1 - BRING RECOGNIZER(S) ONLINE");
    PUT_LINE (ITEM => "   2 - INPUT FILE FOR RECOGNITION TEST");
    PUT_LINE (ITEM => "   3 - SET RECOGNIZER TYPE (TIME WARP)");
    PUT_LINE (ITEM => "   4 - RECOGNIZE USING LP POWER SPECTRUM");
    PUT_LINE (ITEM => "   5 - RECOGNIZE USING LP DELTA POWER SPECTRUM");
    PUT_LINE (ITEM => "   6 - RECOGNIZE USING LF CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "   7 - RECOGNIZE USING LF DELTA CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "   8 - RECOGNIZE USING MF CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "   9 - RECOGNIZE USING MF DELTA CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "  10 - RECOGNIZE USING BL CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "  11 - RECOGNIZE USING BL DELTA CEPSTRUM COEFFICIENTS");
    PUT_LINE (ITEM => "  12 - RECOGNIZE USING ALL FEATURES");
    PUT_LINE (ITEM => "  13 - RECOGNIZE USING FUSED FEATURES");
    PUT_LINE (ITEM => "  14 - OUTPUT ERROR SURFACE");
    PUT_LINE (ITEM => "  15 - QUIT");
```

```
NEW_LINE;
PUT (ITEM => "    CHOICE -> ");
CHOICE_INTEGER_IO.GET (ITEM => CHOICE);
GET_LINE (ITEM => DUMMY);
case CHOICE is
  when 1 =>
NEW_LINE;
PUT_LINE (ITEM => "ALL RECOGNIZERS?");
GET (ITEM => ONLINE);
GET_LINE (ITEM => DUMMY);
if ONLINE = 'Y' OR ONLINE = 'y' then
  LPPS_SPEECH.UPDATE_WORD_LIST;
  DPPS_SPEECH.UPDATE_WORD_LIST;
  LFCC_SPEECH.UPDATE_WORD_LIST;
  DFCC_SPEECH.UPDATE_WORD_LIST;
  MFMC_SPEECH.UPDATE_WORD_LIST;
  DFMC_SPEECH.UPDATE_WORD_LIST;
  BLMC_SPEECH.UPDATE_WORD_LIST;
  DLMC_SPEECH.UPDATE_WORD_LIST;
else
  NEW_LINE;
  PUT_LINE (ITEM => "BRING RECOGNIZER LPPS ONLINE?");
  GET (ITEM => ONLINE);
  GET_LINE (ITEM => DUMMY);
  if ONLINE = 'Y' OR ONLINE = 'y' then
LPPS_SPEECH.UPDATE_WORD_LIST;
  end if;
  PUT_LINE (ITEM => "BRING RECOGNIZER DPPS ONLINE?");
  GET (ITEM => ONLINE);
  GET_LINE (ITEM => DUMMY);
  if ONLINE = 'Y' OR ONLINE = 'y' then
DPPS_SPEECH.UPDATE_WORD_LIST;
  end if;
  PUT_LINE (ITEM => "BRING RECOGNIZER LFCC ONLINE?");
  GET (ITEM => ONLINE);
  GET_LINE (ITEM => DUMMY);
  if ONLINE = 'Y' OR ONLINE = 'y' then
LFCC_SPEECH.UPDATE_WORD_LIST;
  end if;
  PUT_LINE (ITEM => "BRING RECOGNIZER DFCC ONLINE?");
  GET (ITEM => ONLINE);
  GET_LINE (ITEM => DUMMY);
```

```
   if ONLINE = 'Y' OR ONLINE = 'y' then
 DFCC_SPEECH.UPDATE_WORD_LIST;
   end if;
   PUT_LINE (ITEM => "BRING RECOGNIZER MFMC ONLINE?");
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = 'Y' OR ONLINE = 'y' then
 MFMC_SPEECH.UPDATE_WORD_LIST;
   end if;
   PUT_LINE (ITEM => "BRING RECOGNIZER DFMC ONLINE?");
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = 'Y' OR ONLINE = 'y' then
 DFMC_SPEECH.UPDATE_WORD_LIST;
   end if;
   PUT_LINE (ITEM => "BRING RECOGNIZER BLMC ONLINE?");
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = 'Y' OR ONLINE = 'y' then
 BLMC_SPEECH.UPDATE_WORD_LIST;
   end if;
   PUT_LINE (ITEM => "BRING RECOGNIZER DLMC ONLINE?");
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = 'Y' OR ONLINE = 'y' then
 DLMC_SPEECH.UPDATE_WORD_LIST;
   end if;
 end if;
   when 2 =>
 NEW_LINE;
 PUT_LINE (ITEM => "TYPE IN THE NAME OF THE FILE FOR RECOGNITION, LESS THE EX-
TENSION");
 GET_LINE (ITEM => THE_FILE);
   when 3 =>
 NEW_LINE;
 PUT_LINE (ITEM => "TYPE IN TIME_WARP, TWO_DEE, ERROR");
 GET (ITEM => RECOGNIZER);
          GET_LINE (ITEM => DUMMY);
   when 4 =>
          if RECOGNIZER = TIME_WARP then
   RESULTS := LPPS_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LPPS_TW");
```

```
          elsif RECOGNIZER = ERROR then
            RESULTS := LPPS_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LPPS.ER");
          else
   RESULTS := LPPS_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LPPS_2D");
          end if;
 ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   when 5 =>
          if RECOGNIZER = TIME_WARP then
   RESULTS := DPPS_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DPPS_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := DPPS_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DPPS.ER");
          else
   RESULTS := DPPS_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DPPS_2D");
          end if;
 ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   when 6 =>
          if RECOGNIZER = TIME_WARP then
   RESULTS := LFCC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LFCC_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := LFCC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LFCC.ER");
          else
   RESULTS := LFCC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LFCC_2D");
          end if;
 ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   when 7 =>
          if RECOGNIZER = TIME_WARP then
   RESULTS := DFCC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DFCC_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := DFCC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DFCC.ER");
          else
   RESULTS := DFCC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DFCC_2D");
```

```
          end if;
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
  when 8 =>
          if RECOGNIZER = TIME_WARP then
  RESULTS := MFMC_SPEECH.RECOGNIZE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"MFMC_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := MFMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"MFMC.ER");
          else
  RESULTS := MFMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"MFMC_2D");
          end if;
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
  when 9 =>
          if RECOGNIZER = TIME_WARP then
  RESULTS := DFMC_SPEECH.RECOGNIZE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"DFMC_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := DFMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"DFMC.ER");
          else
  RESULTS := DFMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"DFMC_2D");
          end if;
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
  when 10 =>
          if RECOGNIZER = TIME_WARP then
  RESULTS := BLMC_SPEECH.RECOGNIZE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"BLMC_TW");
          elsif RECOGNIZER = ERROR then
            RESULTS := BLMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"BLMC.ER");
          else
  RESULTS := BLMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"BLMC_2D");
          end if;
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
  when 11 =>
          if RECOGNIZER = TIME_WARP then
  RESULTS := DLMC_SPEECH.RECOGNIZE(THE_FILE);
  ASSIGN(RESULTS.FEATURE_NAME,"DLMC_TW");
```

```
         elsif RECOGNIZER = ERROR then
           RESULTS := DLMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DLMC.ER");
         else
   RESULTS := DLMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DLMC_2D");
         end if;
 ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   when 12 =>
         if RECOGNIZER = TIME_WARP then
   RESULTS := LPPS_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LPPS_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := DPPS_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DPPS_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := LFCC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LFCC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := DFCC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DFCC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := MFMC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"MFMC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := DFMC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DFMC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := BLMC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"BLMC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS:= DLMC_SPEECH.RECOGNIZE(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DLMC_TW");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
         elsif RECOGNIZER = ERROR then
           RESULTS := LPPS_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"LPPS.ER");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := DPPS_SPEECH.RECOGNIZE_ERROR(THE_FILE);
   ASSIGN(RESULTS.FEATURE_NAME,"DPPS_ER");
   ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
   RESULTS := LFCC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
```

```
ASSIGN(RESULTS.FEATURE_NAME,"LFCC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DFCC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DFCC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := MFMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"MFMC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DFMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DFMC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := BLMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"BLMC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DLMC_SPEECH.RECOGNIZE_ERROR(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DLMC_ER");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
        else
RESULTS := LPPS_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"LPPS_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DPPS_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DPPS_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := LFCC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"LFCC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DFCC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DFCC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := MFMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"MFMC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DFMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DFMC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := BLMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"BLMC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
RESULTS := DLMC_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
ASSIGN(RESULTS.FEATURE_NAME,"DLMC_2D");
ADD_TO(RESULTS, FUSION_HEAD, FUSION_TAIL);
```

```
            end if;
        when 13 =>
      COMPONENTS := 0;
      INPUT: loop
        PUT_LINE (ITEM => "TYPE IN NAME OF THE FEATURE");
        GET_LINE (ITEM => NEW_FEATURES.NAME);
        PUT_LINE (ITEM => "TYPE THE NUMBER OF COMPONENTS");
        INTEGER_IO.GET (ITEM => NEW_FEATURES.COMPONENTS);
        COMPONENTS := COMPONENTS + NEW_FEATURES.COMPONENTS;
        GET_LINE (ITEM => DUMMY);
         FUSED_FEATURE_LIST.ADD_TO(NEW_FEATURES,FUSED_FEA-
TURE_HEAD,FUSED_FEATURE_TAIL);
        PUT (ITEM => "DO YOUWANT TO ADD ANOTHER FEATURE? ");
        GET (ITEM => ONLINE);
        GET_LINE (ITEM => DUMMY);
      exit INPUT when ONLINE = 'N' OR ONLINE = 'n';
      end loop INPUT;
      PUT_LINE (ITEM => "TYPE NAME OF NEW FEATURE");
      GET_LINE (ITEM => NEW_FEATURE_NAME);
      declare
        package FUSED_SPEECH is new REC_SUPPORT_PACKAGE
        (NUMBER_OF_COMPONENTS => COMPONENTS,
         NAME_OF_FEATURE => STRING_VALUE(NEW_FEATURE_NAME));
        use FUSED_SPEECH;
      begin
         FUSED_SPEECH.CREATE_FUSION_WORD_LIST(FUSED_FEATURE_HEAD,FUSED_FEA-
TURE_TAIL);
         FUSED_SPEECH.CREATE_FUSION_WORD(THE_FILE,FUSED_FEA-
TURE_HEAD,FUSED_FEATURE_TAIL);
              if RECOGNIZER = TIME_WARP then
          RESULTS := FUSED_SPEECH.RECOGNIZE(THE_FILE);
           RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_TW";
        elsif RECOGNIZER = ERROR then
      RESULTS := FUSED_SPEECH.RECOGNIZE_ERROR(THE_FILE);
           RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_ER";
              else
          RESULTS := FUSED_SPEECH.RECOGNIZE_TWO_DEE(THE_FILE);
           RESULTS.FEATURE_NAME := NEW_FEATURE_NAME&"_2D";
              end if;
        ADD_TO(RESULTS,FUSION_HEAD,FUSION_TAIL);
      end;
        when 14 =>
           PUT_LINE (ITEM => "NAME OF WORD 1");
        GET_LINE (ITEM => WORD1);
        PUT_LINE (ITEM => “NAME OF WORD 2”);
        GET_LINE (ITEM => WORD2);
        PUT_LINE (ITEM => “NAME OF OUTPUT FILE”);
        GET_LINE (ITEM => THE_FILE);
PUT_LINE (ITEM => “ALL FEATURES?”);
GET (ITEM => ONLINE);
GET_LINE (ITEM => DUMMY);
if ONLINE = ‘Y’ OR ONLINE = ‘y’ then
   LPPS_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   DPPS_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   LFCC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   DFCC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   MFMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   DFMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   BLMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   DLMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
else
   NEW_LINE;
   PUT_LINE (ITEM => “OUTPUT FEATURE LPPS?”);
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = ‘Y’ OR ONLINE = ‘y’ then
LPPS_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   end if;
   PUT_LINE (ITEM => “OUTPUT FEATURE DPPS?”);
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = ‘Y’ OR ONLINE = ‘y’ then
DPPS_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   end if;
   PUT_LINE (ITEM => “OUTPUT FEATURE LFCC?”);
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = ‘Y’ OR ONLINE = ‘y’ then
LFCC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
   end if;
   PUT_LINE (ITEM => “OUTPUT FEATURE DFCC?”);
   GET (ITEM => ONLINE);
   GET_LINE (ITEM => DUMMY);
   if ONLINE = ‘Y’ OR ONLINE = ‘y’ then
DFCC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
```

```
      end if;
      PUT_LINE (ITEM => "OUTPUT FEATURE MFMC?");
      GET (ITEM => ONLINE);
      GET_LINE (ITEM => DUMMY);
      if ONLINE = 'Y' OR ONLINE = 'y' then
    MFMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
      end if;
      PUT_LINE (ITEM => "OUTPUT FEATURE DFMC?");
      GET (ITEM => ONLINE);
      GET_LINE (ITEM => DUMMY);
      if ONLINE = 'Y' OR ONLINE = 'y' then
    DFMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
      end if;
      PUT_LINE (ITEM => "OUTPUT FEATURE BLMC?");
      GET (ITEM => ONLINE);
      GET_LINE (ITEM => DUMMY);
      if ONLINE = 'Y' OR ONLINE = 'y' then
    BLMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
      end if;
      PUT_LINE (ITEM => "OUTPUT FEATURE DLMC?");
      GET (ITEM => ONLINE);
      GET_LINE (ITEM => DUMMY);
      if ONLINE = 'Y' OR ONLINE = 'y' then
    DLMC_SPEECH.OUTPUT_ERROR_SURFACE(WORD1,WORD2,THE_FILE);
      end if;
    end if;
      when 15 =>
    exit MAIN;
    end case;
      exception
    when DATA_ERROR =>
      PUT_LINE (ITEM => "THERE ARE 14 CHOICES, GET WITH THE PROGRAM AND TYPE A
NUMBER BETWEEN 1 AND 9");
      end;
    end loop MAIN;
  end RECOGNITION_CONTROL_PANEL;
end RECOGNITION_PACKAGE;
```

```
- [source_file_header_comment]
with TEXT_IO, REC_SUPPORT_PACKAGE, DATABASE_PACKAGE, STRING_OPERATIONS;
with FUSION_REC_PACKAGE, LINKED_LIST;
use TEXT_IO, DATABASE_PACKAGE, STRING_OPERATIONS, FUSION_REC_PACKAGE;
package TEMPLATE_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    This package instantiates the rec_support_package for all the different
-  features for creating templates
-
- [optional package tags]
- -

  M : constant := 16;

  use PREFIX_LIST;

  - instantiate for each feature
  package LPPS_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => 16,
      NAME_OF_FEATURE => "LPPS"
      );
  package DPPS_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => 16,
      NAME_OF_FEATURE => "DPPS"
      );
  package LFCC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "LFCC"
      );
  package DFCC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "DFCC"
      );
  package MFMC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "MFMC"
      );
  package DFMC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "DFMC"
      );
  package BLMC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
      NAME_OF_FEATURE => "BLMC"
      );
  package DLMC_TEMP is new REC_SUPPORT_PACKAGE (NUMBER_OF_COMPONENTS => M,
        NAME_OF_FEATURE => "DLMC"
        );
    use LPPS_TEMP, DPPS_TEMP, LFCC_TEMP, DFCC_TEMP, PREFIX_LIST;
    use MFMC_TEMP, DFMC_TEMP, BLMC_TEMP, DLMC_TEMP;

    - ++
    -
    - FUNCTIONAL DESCRIPTION:
    -
    -     This procedure provides and interactive menu for doing template
    -  operations
    -
    - [formal parameters]
    - [design]
    - [optional subprogram tags]
    - -
    procedure TEMPLATE_CONTROL_PANEL;

end TEMPLATE_PACKAGE;
```

```
- [source_file_header_comment]
package body TEMPLATE_PACKAGE is
- [package_body_header_comment]

  procedure TEMPLATE_CONTROL_PANEL is
  - [procedure_header_comment]
    subtype CHOICE_INTEGER is INTEGER range 1..8;
    CHOICE : CHOICE_INTEGER;
    ONLINE : CHARACTER;
    IN_PREFIX, THE_FILE, DUMMY, PREFIX, OUT_PREFIX : VAR_STRING;
    TEMP_FACTOR : FLOAT := 0.2;

    package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
    package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
    package CHOICE_INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
    use CHOICE_INTEGER_IO, FLOAT_IO, INTEGER_IO;

  begin
    MAIN: loop
      begin
    NEW_LINE(2);
    PUT_LINE (ITEM => "   1 - C, CREATE LIST OF PREFIXES");
    PUT_LINE (ITEM => "   2 - C U, SET WORD NAME FOR TEMPLATING");
    PUT_LINE (ITEM => "   3 - C U, SET OUTPUT TEMPLATE PREFIX");
    PUT_LINE (ITEM => "   4 - U, SET TEMPLATE FACTOR");
    PUT_LINE (ITEM => "   5 - U, SET INPUT TEMPLATE PREFIX");
    PUT_LINE (ITEM => "   6 - CREATE TEMPLATE");
    PUT_LINE (ITEM => "   7 - UPDATE TEMPLATE");
    PUT_LINE (ITEM => "   8 - QUIT TEMPLATE SUBSYSTEM");
    NEW_LINE;
    PUT (ITEM => "    CHOICE -> ");
    CHOICE_INTEGER_IO.GET (ITEM => CHOICE);
    GET_LINE (ITEM => DUMMY);
    case CHOICE is
      when 1 =>
    PREFIX_LIST.CLEAR_LIST(PREFIX_HEAD,PREFIX_TAIL);
    NEW_LINE;
              GET_PREFIXES: loop
                PUT_LINE (ITEM => "TYPE NAME OF PREFIX");
            GET_LINE (ITEM => PREFIX);
      PREFIX_LIST.ADD_TO(PREFIX,PREFIX_HEAD,PREFIX_TAIL);
            PUT_LINE (ITEM => "DO YOU WANT TO ADD ANOTHER?");
    GET(ONLINE);
        GET_LINE (ITEM => DUMMY);
        exit GET_PREFIXES when ONLINE = 'N' OR ONLINE = 'n';
          end loop GET_PREFIXES;
    when 2 =>
  NEW_LINE;
  PUT_LINE (ITEM => "TYPE IN THE NAME OF THE WORD FOR TEMPLATING, THE EXTEN-
SION");
  GET_LINE (ITEM => THE_FILE);
    when 3 =>
  NEW_LINE;
  PUT_LINE (ITEM => "TYPE IN THE NAME OF THE OUTPUT PREFIX");
          GET_LINE (ITEM => OUT_PREFIX);
    when 4 =>
          PUT_LINE (ITEM => "TYPE IN THE TEMPLATING FACTOR");
          GET (ITEM => TEMP_FACTOR);
  GET_LINE (ITEM => DUMMY);
    when 5 =>
  NEW_LINE;
  PUT_LINE (ITEM => "TYPE IN THE NAME OF THE INPUT PREFIX");
          GET_LINE (ITEM => IN_PREFIX);
    when 6 =>
  LPPS_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  DPPS_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  LFCC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  DFCC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  MFMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  DFMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  BLMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
  DLMC_TEMP.CREATE_TW_TEMPLATE(OUT_PREFIX,THE_FILE);
    when 7 =>
  LPPS_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FACTOR);
  DPPS_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FACTOR);
  LFCC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FACTOR);
  DFCC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FAC-
TOR);
  MFMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FAC-
TOR);
  DFMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FAC-
TOR);
  BLMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FAC-
TOR);
  DLMC_TEMP.UPDATE_TW_TEMPLATE(IN_PREFIX,OUT_PREFIX,THE_FILE,TEMP_FAC-
TOR);
```

```
      when 8 =>
    exit MAIN;
    end case;
      exception
    when DATA_ERROR =>
      PUT_LINE (ITEM => “THERE ARE 8 CHOICES, GET WITH THE PROGRAM AND TYPE A
NUMBER BETWEEN 1 AND 9”);
      end;
    end loop MAIN;
  end TEMPLATE_CONTROL_PANEL;
end TEMPLATE_PACKAGE;
```

```
with TEXT_IO, FLOAT_MATH_LIB, STRING_OPERATIONS, DATABASE_PACKAGE,
LINKED_LIST;
with FUSION_REC_PACKAGE;
use TEXT_IO, FLOAT_MATH_LIB, STRING_OPERATIONS, DATABASE_PACKAGE;
use FUSION_REC_PACKAGE;

generic

  ORDER : INTEGER := 5;
  NUMBER_OF_COMPONENTS : INTEGER := 1;
  NAME_OF_FEATURE : STRING;

- ++
-
- FACILITY:
-
-    {tbs}
-
- ABSTRACT:
-
-    This package specifies the speech utterance data types and operations
-    that take place on this data type
-
- AUTHORS:
-
-    THOMAS RATHBUN
-
- CREATION DATE:
-
-    22 Oct 90
-
- MODIFICATION HISTORY:
-
-    MODIFIED FOR THE ASRT PROGRAM, JUL,AUG,SEP 1991
-
- -
package REC_SUPPORT_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    THIS PACKAGE CONTAINS ALL THE PROCEDURES AND FUNCTIONS NECESSARY FOR
- DOING SPEECH RECOGNITION.  THIS PACKAGE IS GENERIC, TO BE INSTANTIATED FOR
- FOR A SPECIFIC FEATURE.
-
- -
  use FUSION_LIST, FUSED_FEATURE_LIST;

  - MATRIX TYPE USED FOR INPUT INTO THE TWO-DIMENTIONAL FFT
  type MATRIX_TYPE is array (INTEGER range <>, INTEGER range <>) of FLOAT;
  - ARRAY TYPE FOR STORING ALL THE COMPONENTS IN A SINGLE TIME SLICE
  type SLICE_ARRAY_TYPE is array (INTEGER RANGE 1..NUMBER_OF_COMPONENTS) of
FLOAT;
  - LIST FOR STORING ALL TIME SLICES IN A WORD
  package SLICE_LIST is new LINKED_LIST (ITEM_TYPE => SLICE_ARRAY_TYPE);
  use SLICE_LIST;

  - RECORD TYPE FOR STORING AN ENTIRE WORD
  type WORD_TYPE is
    record
      WORD_HEAD : SLICE_LIST.LIST;
      WORD_TAIL : SLICE_LIST.LIST;
      WORD_LENGTH : INTEGER := 0;
      WORD_NAME : VAR_STRING;
    end record;

  FILTER : INTEGER := ORDER * 2 + 1;

  - LIST FOR STORING ALL THE WORDS IN THE DATABASE
  package WORD_LIST is new LINKED_LIST (ITEM_TYPE => WORD_TYPE);
  use WORD_LIST;

  package INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
  package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
  use FLOAT_IO, INTEGER_IO;

  - ++
  - FUNCTIONAL DESCRIPTION:
  -
  -    THIS FUNCTION PERFORMS DYNAMIC TIME WARPING BETWEEN AN UNKNOWN
  -  WORD AND THE WORDS IN THE DATABASE LIST
  -
  - [formal parameters]
  - [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    RETURNS A RECORD CONTAINING A LIST OF DTW DISTANCE
-
--
function RECOGNIZE (NAME : in VAR_STRING) return RESULT_RECORD_TYPE;

-++
- FUNCTIONAL DESCRIPTION:
-
-    THIS FUNCTION PERFORMS LOW FREQUENCY TWO-DIMENSIONAL FFT COMPARISON
-  ON THE IMAGE OF AN UNKNOWN WORDS AND THE WORDS INTHE DATABASE LIST
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    RETURNS A RECORD CONTAINING A LIST OF LOW FREQUNCY DISTANCES
-
--
function RECOGNIZE_TWO_DEE (NAME : in VAR_STRING) return RESULT_RECORD_TYPE;

-++
- FUNCTIONAL DESCRIPTION:
-
-    THIS FUNCTION PERFORMS A LOW FREQUNCY TWO-DIMENSIONAL FFT BETWEEN
-  THE ERROR SURFACE OF AN UNKNWON WORD AND THE WORDS IN THE DATABASE
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    RETURNS A RECORD CONTAINING A LIST OF LOW FREQUNCY DISTANCES
-
--
function RECOGNIZE_ERROR (NAME : in VAR_STRING) return RESULT_RECORD_TYPE;

-++
-
```

```
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE READS ALL FILES OF THE WORDS IN THE DATABASE AND
-  STORES THEM IN A LIST FORRECOGNITION PURPOSES
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure UPDATE_WORD_LIST;

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURES CREATES A NEW DATABASE WORD LIST FOR A FUSED FEATURE
-  FORM THE FEATURE WORDS LIST ALREADY CREATED
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure CREATE_FUSION_WORD_LIST (FUSION_FEATURE_HEAD, FUSION_FEA-
TURE_TAIL : in out FUSED_FEATURE_LIST.LIST);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE CREATES A NEW FEATURE FILE FOR THE UNKNOWN WORD
-  FROM THE FEATURE FILES ON DISK
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure CREATE_FUSION_WORD (WORD_NAME : in VAR_STRING;
   FUSION_FEATURE_HEAD, FUSION_FEATURE_TAIL : in out FUSED_FEATURE_LIST.LIST);

- ++
- FUNCTIONAL DESCRIPTION:
-
```

```
-    This function takes two words and returns a new word where the new
- word is has been time warp aligned from the known word to the unknown
- word
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    {tbs}
-
- -
 function TIME_WARP_ALIGNMENT (KNOWN_WORD, UNKNOWN_WORD : in WORD_TYPE)
return WORD_TYPE;


 - ++
 -
 - FUNCTIONAL DESCRIPTION:
 -
 -    This procedure creates an error surface and writes it to a file
 -
 - [formal parameters]
 - [design]
 - [optional subprogram tags]
 - -
 procedure OUTPUT_ERROR_SURFACE (WORD_NAME1, WORD_NAME2, FILE_NAME : in
VAR_STRING);


 - ++
 -
 - FUNCTIONAL DESCRIPTION:
 -
 -    This procedure creates a template from a list of prefixes using
 - time warp alignment
 -
 - [formal parameters]
 - [design]
 - [optional subprogram tags]
 - -
 procedure CREATE_TW_TEMPLATE (PREFIX_OUT, WORD_NAME : in VAR_STRING);


 - ++
 -
```

```
  - FUNCTIONAL DESCRIPTION:
  -
  -    This procedure updates a new set of words into a template
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure UPDATE_TW_TEMPLATE (PREFIX_IN, PREFIX_OUT, WORD_NAME : in
VAR_STRING;
     TEMPLATE_FACTOR : in FLOAT);

  LIBRARY_HEAD, LIBRARY_TAIL : WORD_LIST.LIST;
  TEMPLATE_HEAD, TEMPLATE_TAIL : WORD_LIST.LIST;

end REC_SUPPORT_PACKAGE;
```

```
package body REC_SUPPORT_PACKAGE is
- ++
-
- DESIGN:
-
-    {tbs}
-
- -
     TEST_FILE : FILE_TYPE;

  procedure GET (WORD : in out WORD_TYPE; FILE : in VAR_STRING) is
  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    {tbs}
  -
  - FORMAL PARAMETERS:
  -
  -    {subtags}
  -
  - DESIGN:
  -
  -    {tbs}
  -
  - -
     THE_FILE : FILE_TYPE;
     COUNTER : INTEGER := 0;
     A_SLICE : SLICE_ARRAY_TYPE;

  begin
     OPEN (
        FILE => THE_FILE,
        MODE => IN_FILE,
        NAME => FILE);
     CLEAR_LIST(WORD.WORD_HEAD,WORD.WORD_TAIL);
     while NOT END_OF_FILE (FILE => THE_FILE) loop
        COUNTER := COUNTER + 1;
        for INDEX in 1..NUMBER_OF_COMPONENTS loop
           GET (FILE => THE_FILE, ITEM => A_SLICE(INDEX));
        end loop;
        ADD_TO(A_SLICE,WORD.WORD_HEAD,WORD.WORD_TAIL);
  end loop;
  CLOSE (FILE => THE_FILE);
  WORD.WORD_LENGTH := COUNTER;
end GET;

procedure PUT (WORD : in WORD_TYPE; FILE_NAME : in VAR_STRING) is
- [procedure_header_comment]
  OUTPUT_FILE : FILE_TYPE;
  POINTER : SLICE_LIST.LIST := WORD.WORD_HEAD;
  A_SLICE : SLICE_ARRAY_TYPE;
begin
  CREATE (
    FILE => OUTPUT_FILE,
    MODE => OUT_FILE,
    NAME => FILE_NAME);
  loop
    A_SLICE := GET_NODE(POINTER);
    for INDEX in 1..NUMBER_OF_COMPONENTS loop
      PUT (FILE => OUTPUT_FILE, ITEM => A_SLICE(INDEX),
        FORE => 6,
        AFT => 4,
        EXP => 0);
  NEW_LINE (FILE => OUTPUT_FILE);
    end loop;
    POINTER := NEXT_LINK(POINTER);
  end loop;
exception
  when SLICE_LIST.UNDER_FLOW =>
    CLOSE (FILE => OUTPUT_FILE);
end PUT;

function TWO_DEE_EUCLIDIAN (WORD1, WORD2 : in MATRIX_TYPE) return FLOAT is
- [function_header_comment]
  SUM : FLOAT := 0.0;
begin
  for ROW in 0..FILTER-1 loop
    for COLUMN in 0..FILTER-1 loop
      SUM := SUM + (WORD1(ROW,COLUMN)-WORD2(ROW,COLUMN))**2;
    end loop;
  end loop;
  return SQRT(SUM);
end TWO_DEE_EUCLIDIAN;
```

```
function EUCLIDIAN_DISTANCE (WORD1, WORD2 : in SLICE_ARRAY_TYPE) return FLOAT is
- [function_header_comment]
  SUM : FLOAT := 0.0;
begin
  for INDEX in 1..NUMBER_OF_COMPONENTS loop
    SUM := SUM + (WORD1(INDEX) - WORD2(INDEX))**2;
  end loop;
  return SQRT(SUM);
end EUCLIDIAN_DISTANCE;

function TIME_WARP (KNOWN_WORD, UNKNOWN_WORD : in WORD_TYPE) return FLOAT is
- [function_header_comment]

  JUMP_UP, JUMP_OVER, JUMP_DIAGONAL, JUMP_UP_DIAGONAL : FLOAT;
  UP_POINTER, RIGHT_POINTER : SLICE_LIST.LIST;
  DISTANCE : FLOAT := 0.0;
  NO_OVER : BOOLEAN := FALSE;

begin
  SLICE_LIST.NULL_POINTER(UP_POINTER);
  SLICE_LIST.NULL_POINTER(RIGHT_POINTER);
  UP_POINTER := KNOWN_WORD.WORD_HEAD;
  RIGHT_POINTER := UNKNOWN_WORD.WORD_HEAD;
  DISTANCE := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
  RIGHT_POINTER := NEXT_LINK(UNKNOWN_WORD.WORD_HEAD);
   begin
    HORIZONTAL: loop
  if NO_OVER then
    UP_POINTER := NEXT_LINK(UP_POINTER);
    JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
    JUMP_UP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
    if JUMP_DIAGONAL < JUMP_UP_DIAGONAL then
  DISTANCE := DISTANCE + JUMP_DIAGONAL;
    else
  DISTANCE := DISTANCE + JUMP_UP_DIAGONAL;
  UP_POINTER := NEXT_LINK(UP_POINTER);
    end if;
    NO_OVER := FALSE;
  else
```

```
        JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
        JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
        JUMP_UP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(NEXT_LINK(UP_POINTER))));
        if JUMP_OVER < JUMP_DIAGONAL AND JUMP_OVER < JUMP_UP_DIAGONAL then
      DISTANCE := DISTANCE + JUMP_OVER;
      NO_OVER := TRUE;
        elsif JUMP_DIAGONAL < JUMP_OVER AND JUMP_DIAGONAL < JUMP_UP_DIAGONAL
then
      DISTANCE := DISTANCE + JUMP_DIAGONAL;
      UP_POINTER := NEXT_LINK(UP_POINTER);
        else
      DISTANCE := DISTANCE + JUMP_UP_DIAGONAL;
      UP_POINTER := NEXT_LINK(NEXT_LINK(UP_POINTER));
        end if;
      end if;
        exit HORIZONTAL when IS_NULL(NEXT_LINK(RIGHT_POINTER));
      RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
        end loop HORIZONTAL;
-        begin
-           loop
-       UP_POINTER := NEXT_LINK(UP_POINTER);
-              JUMP_UP := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
-       DISTANCE := DISTANCE + JUMP_UP;
-           end loop;
-        exception
-        PUT (ITEM => UNKNOWN_WORD.WORD_LENGTH);
-        PUT (ITEM => KNOWN_WORD.WORD_LENGTH);
-        PUT (ITEM => DISTANCE,
-           FORE => 4,
-           AFT => 4,
-           EXP => 0);
-       NEW_LINE;
-           when SLICE_LIST.UNDER_FLOW =>
        return DISTANCE;
-        end;
       exception
         when SLICE_LIST.UNDER_FLOW =>
           if IS_NULL(NEXT_LINK(UP_POINTER)) then
        DISTANCE := DISTANCE + JUMP_OVER;
        RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
```

```
            while NOT IS_NULL(RIGHT_POINTER) loop
              JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
      DISTANCE := DISTANCE + JUMP_OVER;
      RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
            end loop;
          else
            GET_UP_LOOP: loop
      JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
      JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
      if JUMP_OVER < JUMP_DIAGONAL then
        DISTANCE := DISTANCE + JUMP_OVER;
      else
        DISTANCE := DISTANCE + JUMP_DIAGONAL;
        UP_POINTER := NEXT_LINK(UP_POINTER);
      end if;
      RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
               exit GET_UP_LOOP when IS_NULL(RIGHT_POINTER) OR
IS_NULL(NEXT_LINK(UP_POINTER));
            end loop GET_UP_LOOP;
            FINSHISH_LOOP: while NOT IS_NULL(RIGHT_POINTER) loop
      while NOT IS_NULL(RIGHT_POINTER) loop
        JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
        DISTANCE := DISTANCE + JUMP_OVER;
        RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
      end loop;
            end loop FINSHISH_LOOP;
          end if;
-         PUT (ITEM => UNKNOWN_WORD.WORD_LENGTH);
-         PUT (ITEM => KNOWN_WORD.WORD_LENGTH);
-         PUT (ITEM => DISTANCE,
-           FORE => 4,
-           AFT => 4,
-           EXP => 0);
-        NEW_LINE;
         return DISTANCE;
      end;
    end TIME_WARP;

    function RECOGNIZE (NAME : in VAR_STRING) return RESULT_RECORD_TYPE is
    - [function_header_comment]
    UNKNOWN_WORD, KNOWN_WORD : WORD_TYPE;
    WORD_NAME, THIS_FEATURE, UNKNOWN_WORD_NAME : VAR_STRING;
    WORD_POINTER : WORD_LIST.LIST := LIBRARY_HEAD;
    WORD_NAME_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
    DTW_RESULTS : RESULT_RECORD_TYPE;
    WORD_COUNT, SMALLEST : INTEGER := 0;
    DISTANCE, NORMAL : FLOAT := 0.0;

  begin
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_TAIL);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_TAIL);
    RESULT_LIST.NULL_POINTER(DTW_RESULTS.RESULT_HEAD);
    RESULT_LIST.NULL_POINTER(DTW_RESULTS.RESULT_TAIL);
    ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
    UNKNOWN_WORD_NAME := DIRECTORY_NAME&NAME&"."&THIS_FEATURE;
    GET(UNKNOWN_WORD,UNKNOWN_WORD_NAME);
    RESULT_LIST.CLEAR_LIST(DTW_RESULTS.RESULT_HEAD,DTW_RESULTS.RESULT_TAIL);
    begin
      loop
    NEXT_WORD(WORD_NAME,WORD_NAME_POINTER);
    WORD_COUNT := WORD_COUNT + 1;
    GET_WORD: loop
      KNOWN_WORD := GET_NODE(WORD_POINTER);
      WORD_POINTER := NEXT_LINK(WORD_POINTER);
    exit GET_WORD when IS_EQUAL(KNOWN_WORD.WORD_NAME,WORD_NAME);
    end loop GET_WORD;
    DISTANCE := TIME_WARP(KNOWN_WORD, UNKNOWN_WORD);
    RESULT_LIST.ADD_TO(DISTANCE,DTW_RESULTS.RESULT_HEAD,DTW_RESULTS.RE-
SULT_TAIL);
    NORMAL := NORMAL + DISTANCE**2;
      end loop;
    exception
      when DATABASE_LIST.UNDER_FLOW =>
        NULL;
    end;
    NORMAL := SQRT(NORMAL);
    for INDEX in 1..WORD_COUNT loop
      RESULT_LIST.REMOVE_NEXT_NODE(DISTANCE,DTW_RESULTS.RE-
SULT_HEAD,DTW_RESULTS.RESULT_TAIL);
      DISTANCE := DISTANCE/NORMAL;
      RESULT_LIST.ADD_TO(DISTANCE,DTW_RESULTS.RESULT_HEAD,DTW_RESULTS.RE-
SULT_TAIL);
    end loop;
    return DTW_RESULTS;
  end RECOGNIZE;

  procedure TWO_DEE_FFT (IMAGE : in MATRIX_TYPE;
      HEIGHT, WIDTH : in INTEGER;
      COEFF : in out MATRIX_TYPE) is
  - ++
  - FUNCTIONAL DESCRIPTION:
  -
  -    {tbs}
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - RETURN VALUE:
  -
  -    {tbs}
  -
  - -
    NORM, COSSIN_TERM, HIGH, TEMPK, TEMPL, ACROSS, DC_TERM : FLOAT;
    PI : FLOAT := 3.1415;
  begin
    HIGH := 2.0 * PI / FLOAT(HEIGHT);
    ACROSS := 2.0 * PI / FLOAT(WIDTH);
     for INDEX in 0..FILTER-1 loop
       for INDEX1 in 0..FILTER-1 loop
    COEFF(INDEX,INDEX1) := 0.0;
       end loop;
     end loop;
     for K in 0..ORDER-1 loop
       for L in 0..FILTER-1 loop
         TEMPK := FLOAT(K-ORDER)*HIGH;
    TEMPL := FLOAT(L-ORDER)*ACROSS;
         for I in 0..HEIGHT-1 loop
           for J in 0..WIDTH-1 loop
             COSSIN_TERM := FLOAT(I)*TEMPK-FLOAT(J)*TEMPL;
    COEFF(K,L) := COEFF(K,L) + IMAGE(I+1,J+1) * COS(COSSIN_TERM);
    COEFF(FILTER-1-K,L) := COEFF(FILTER-1-K,L) + IMAGE(I+1,J+1) * SIN(COSSIN_TERM);
           end loop;
```

```
          end loop;
        end loop;
      end loop;
      for L in 0..ORDER-1 loop
        TEMPL := FLOAT(L-ORDER) * ACROSS;
        for I in 0..HEIGHT-1 loop
          for J in 0..WIDTH-1 loop
            COSSIN_TERM := -FLOAT(J) * TEMPL;
        COEFF(ORDER,L) := COEFF(ORDER,L) + IMAGE(I+1,J+1) * COS(COSSIN_TERM);
        COEFF(ORDER,FILTER-1-L) := COEFF(ORDER,FILTER-1-L) + IMAGE(I+1,J+1) *
SIN(COSSIN_TERM);
          end loop;
        end loop;
      end loop;
      DC_TERM := 0.0;
      for I in 0..HEIGHT-1 loop
        for J in 0..WIDTH-1 loop
          if IMAGE(I+1,J+1) = 1.0 then
            DC_TERM := DC_TERM + 1.0;
          end if;
        end loop;
      end loop;
      COEFF(ORDER,ORDER) := DC_TERM;
      NORM := 0.0;
      for K in 0..FILTER-1 loop
        for L in 0..FILTER-1 loop
          NORM := NORM + COEFF(K,L)**2;
        end loop;
      end loop;
      NORM := SQRT(NORM);
      for K in 0..FILTER-1 loop
        for L in 0..FILTER-1 loop
          COEFF(K,L) := COEFF(K,L)/NORM;
        end loop;
      end loop;
    end TWO_DEE_FFT;

    function RECOGNIZE_TWO_DEE (NAME : in VAR_STRING) return RESULT_RECORD_TYPE is
    - [function_header_comment]

      UNKNOWN_LF_MATRIX, KNOWN_LF_MATRIX : MATRIX_TYPE(0..FILTER-1,0..FILTER-1);
      UNKNOWN_WORD, KNOWN_WORD : WORD_TYPE;
    WORD_NAME, THIS_FEATURE, UNKNOWN_WORD_NAME : VAR_STRING;
    WORD_POINTER : WORD_LIST.LIST := LIBRARY_HEAD;
    WORD_NAME_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
    SLICE_POINTER : SLICE_LIST.LIST;
    A_TIME_SLICE : SLICE_ARRAY_TYPE;
    TDD_RESULTS : RESULT_RECORD_TYPE;
    DISTANCE, NORMAL : FLOAT := 0.0;
    WORD_COUNT, SMALLEST, SLICE_COUNTER : INTEGER := 0;
  begin
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_TAIL);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_TAIL);
    RESULT_LIST.NULL_POINTER(TDD_RESULTS.RESULT_HEAD);
    RESULT_LIST.NULL_POINTER(TDD_RESULTS.RESULT_TAIL);
    SLICE_LIST.NULL_POINTER(SLICE_POINTER);
    ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
    UNKNOWN_WORD_NAME := DIRECTORY_NAME&NAME&"."&THIS_FEATURE;
    GET(UNKNOWN_WORD,UNKNOWN_WORD_NAME);
    declare
      NEW_WORD : MATRIX_TYPE(1..NUMBER_OF_COMPONENTS,1..UN-
KNOWN_WORD.WORD_LENGTH);
    begin
      begin
    SLICE_COUNTER := 1;
    SLICE_POINTER := UNKNOWN_WORD.WORD_HEAD;
         loop
           A_TIME_SLICE := GET_NODE(SLICE_POINTER);
           for INDEX in 1..NUMBER_OF_COMPONENTS loop
    NEW_WORD(INDEX,SLICE_COUNTER) := A_TIME_SLICE(INDEX);
           end loop;
      SLICE_POINTER := NEXT_LINK(SLICE_POINTER);
      SLICE_COUNTER := SLICE_COUNTER + 1;
         end loop;
      exception
         when SLICE_LIST.UNDER_FLOW =>
           NULL;
      end;
      TWO_DEE_FFT(NEW_WORD,NUMBER_OF_COMPONENTS,UN-
KNOWN_WORD.WORD_LENGTH,UNKNOWN_LF_MATRIX );
    end;
    begin
      loop
    NEXT_WORD(WORD_NAME,WORD_NAME_POINTER);
    WORD_COUNT := WORD_COUNT + 1;
    GET_WORD: loop
      KNOWN_WORD := GET_NODE(WORD_POINTER);
      WORD_POINTER := NEXT_LINK(WORD_POINTER);
    exit GET_WORD when IS_EQUAL(KNOWN_WORD.WORD_NAME,WORD_NAME);
    end loop GET_WORD;
    KNOWN_WORD := TIME_WARP_ALIGNMENT(KNOWN_WORD, UNKNOWN_WORD);
    declare
      NEW_WORD : MATRIX_TYPE(1..NUMBER_OF_COMPO-
NENTS,1..KNOWN_WORD.WORD_LENGTH);
    begin
      begin
    SLICE_COUNTER := 1;
    SLICE_POINTER := KNOWN_WORD.WORD_HEAD;
    loop
      A_TIME_SLICE := GET_NODE(SLICE_POINTER);
      for INDEX in 1..NUMBER_OF_COMPONENTS loop
    NEW_WORD(INDEX,SLICE_COUNTER) := A_TIME_SLICE(INDEX);
      end loop;
      SLICE_POINTER := NEXT_LINK(SLICE_POINTER);
      SLICE_COUNTER := SLICE_COUNTER + 1;
    end loop;
      exception
    when SLICE_LIST.UNDER_FLOW =>
      NULL;
      end;
      TWO_DEE_FFT(NEW_WORD,NUMBER_OF_COMPO-
NENTS,KNOWN_WORD.WORD_LENGTH,KNOWN_LF_MATRIX );
    end;
    DISTANCE := TWO_DEE_EUCLIDIAN(KNOWN_LF_MATRIX,UNKNOWN_LF_MATRIX);
    RESULT_LIST.ADD_TO(DISTANCE,TDD_RESULTS.RESULT_HEAD,TDD_RESULTS.RE-
SULT_TAIL);
    NORMAL := NORMAL + DISTANCE**2;
      end loop;
     exception
       when DATABASE_LIST.UNDER_FLOW =>
         NULL;
     end;
    NORMAL := SQRT(NORMAL);
    for INDEX in 1..WORD_COUNT loop
      RESULT_LIST.REMOVE_NEXT_NODE(DISTANCE,TDD_RESULTS.RE-
SULT_HEAD,TDD_RESULTS.RESULT_TAIL);
```

```
        DISTANCE := DISTANCE/NORMAL;
        RESULT_LIST.ADD_TO(DISTANCE,TDD_RESULTS.RESULT_HEAD,TDD_RESULTS.RE-
SULT_TAIL);
      end loop;
      return TDD_RESULTS;
   end RECOGNIZE_TWO_DEE;

   procedure UPDATE_WORD_LIST is
   - [procedure_header_comment]
      COUNT : INTEGER := 0;
      THE_WORD : WORD_TYPE;
      FILE_NAME, PERIOD, NEXT_WORD_NAME, THIS_FEATURE : VAR_STRING;
      WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
   begin
      CLEAR_LIST(LIBRARY_HEAD,LIBRARY_TAIL);
      SLICE_LIST.NULL_POINTER(THE_WORD.WORD_HEAD);
      SLICE_LIST.NULL_POINTER(THE_WORD.WORD_TAIL);
      ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
      loop
         NEXT_WORD(NEXT_WORD_NAME,WORD_POINTER);
         COUNT := COUNT + 1;
         FILE_NAME := DIRECTORY_NAME&LIBRARY_PRE-
FIX&NEXT_WORD_NAME&"."&THIS_FEATURE;
         GET(THE_WORD,FILE_NAME);
         THE_WORD.WORD_NAME := NEXT_WORD_NAME;
         ADD_TO(THE_WORD,LIBRARY_HEAD,LIBRARY_TAIL);
      end loop;
   exception
      when DATABASE_LIST.UNDER_FLOW =>
         SET_WORD_COUNT(COUNT);
   end UPDATE_WORD_LIST;

   procedure GET (WORD : in out WORD_TYPE; FILE : in VAR_STRING;
        COMPONENTS, COMPONENTS_POINTER : INTEGER;
        FIRST_PASS : BOOLEAN) is
--
      THE_FILE : FILE_TYPE;
      COUNTER : INTEGER := 0;
      A_SLICE : SLICE_ARRAY_TYPE;
   begin
      OPEN (
         FILE => THE_FILE,
         MODE => IN_FILE,
      NAME => FILE);
    loop
      if NOT FIRST_PASS then
    REMOVE_NEXT_NODE(A_SLICE,WORD.WORD_HEAD,WORD.WORD_TAIL);
      end if;
      COUNTER := COUNTER + 1;
      for INDEX in COMPONENTS_POINTER..COMPONENTS_POINTER+COMPONENTS-1 loop
    GET (FILE => THE_FILE, ITEM => A_SLICE(INDEX));
      end loop;
      ADD_TO(A_SLICE,WORD.WORD_HEAD,WORD.WORD_TAIL);
    end loop;
  exception
    when end_error =>
      CLOSE (FILE => THE_FILE);
      WORD.WORD_LENGTH := COUNTER;
  end GET;

  procedure CREATE_FUSION_WORD_LIST (FUSION_FEATURE_HEAD, FUSION_FEA-
TURE_TAIL : in out FUSED_FEATURE_LIST.LIST) is
  - [procedure_header_comment]
    COUNT, COMPONENTS_COUNT : INTEGER;
    WORD_POINTER : DATABASE_LIST.LIST;
    FEATURE_POINTER : FUSED_FEATURE_LIST.LIST := FUSION_FEATURE_HEAD;
    FEATURE_VARIABLE : FEATURE_RECORD_TYPE;
    THIS_FEATURE, NEXT_FEATURE_NAME, NEXT_WORD_NAME, FILE_NAME :
VAR_STRING;
    FUSED_WORD : WORD_TYPE;
    FUSED_FILE : FILE_TYPE;
  begin
    SLICE_LIST.NULL_POINTER(FUSED_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(FUSED_WORD.WORD_TAIL);
    DATABASE_LIST.NULL_POINTER(WORD_POINTER);
    ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
    COMPONENTS_COUNT := 1;
    FEATURE_VARIABLE := FUSED_FEATURE_LIST.GET_NODE(FEATURE_POINTER);
    WORD_POINTER := DATABASE_HEAD;
    COUNT := 0;
     begin
       loop
    CLEAR_LIST(FUSED_WORD.WORD_HEAD,FUSED_WORD.WORD_TAIL);
    NEXT_WORD(NEXT_WORD_NAME,WORD_POINTER);
    COUNT := COUNT + 1;
```

```
     FILE_NAME := DIRECTORY_NAME&LIBRARY_PREFIX&NEXT_WORD_NAME&"."&FEA-
TURE_VARIABLE.NAME;
          GET(FUSED_WORD,FILE_NAME,FEATURE_VARIABLE.COMPONENTS,
       COMPONENTS_COUNT,TRUE);
     ADD_TO(FUSED_WORD,LIBRARY_HEAD,LIBRARY_TAIL);
        end loop;
     exception
        when DATABASE_LIST.UNDER_FLOW =>
          COMPONENTS_COUNT := COMPONENTS_COUNT + FEATURE_VARIABLE.COMPO-
NENTS;
     FEATURE_POINTER := FUSED_FEATURE_LIST.NEXT_LINK(FEATURE_POINTER);
     end;
     loop
        FEATURE_VARIABLE := FUSED_FEATURE_LIST.GET_NODE(FEATURE_POINTER);
        WORD_POINTER := DATABASE_HEAD;
        begin
          loop
        CLEAR_LIST(FUSED_WORD.WORD_HEAD,FUSED_WORD.WORD_TAIL);
        NEXT_WORD(NEXT_WORD_NAME,WORD_POINTER);
        FILE_NAME := DIRECTORY_NAME&LIBRARY_PREFIX&NEXT_WORD_NAME&"."&FEA-
TURE_VARIABLE.NAME;
        REMOVE_NEXT_NODE(FUSED_WORD,LIBRARY_HEAD,LIBRARY_TAIL);
        GET(FUSED_WORD,FILE_NAME,FEATURE_VARIABLE.COMPONENTS,
     COMPONENTS_COUNT,FALSE);
        ADD_TO(FUSED_WORD,LIBRARY_HEAD,LIBRARY_TAIL);
          end loop;
        exception
          when DATABASE_LIST.UNDER_FLOW =>
            COMPONENTS_COUNT := COMPONENTS_COUNT+FEATURE_VARIABLE.COMPO-
NENTS;
        FEATURE_POINTER := FUSED_FEATURE_LIST.NEXT_LINK(FEATURE_POINTER);
        end;
     end loop;
   exception
     when FUSED_FEATURE_LIST.UNDER_FLOW =>
       SET_WORD_COUNT(COUNT);
   end CREATE_FUSION_WORD_LIST;

   procedure CREATE_FUSION_WORD (WORD_NAME : in VAR_STRING;
       FUSION_FEATURE_HEAD, FUSION_FEATURE_TAIL : in out FUSED_FEATURE_LIST.LIST)
is
   - {procedure_header_comment}
     COUNT, COMPONENTS_COUNT : INTEGER;
     FEATURE_POINTER : FUSED_FEATURE_LIST.LIST := FUSION_FEATURE_HEAD;
```

```
    FEATURE_VARIABLE : FEATURE_RECORD_TYPE;
    THIS_FEATURE, NEXT_FEATURE_NAME, FILE_NAME : VAR_STRING;
    FUSED_WORD : WORD_TYPE;
    FUSED_FILE : FILE_TYPE;
  begin
    SLICE_LIST.NULL_POINTER(FUSED_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(FUSED_WORD.WORD_TAIL);
    COMPONENTS_COUNT := 1;
    FEATURE_VARIABLE := FUSED_FEATURE_LIST.GET_NODE(FEATURE_POINTER);
     PUT_LINE (ITEM => WORD_NAME);
     PUT_LINE (ITEM => FEATURE_VARIABLE.NAME);
    FILE_NAME := DIRECTORY_NAME&LIBRARY_PREFIX&WORD_NAME&"."&FEA-
TURE_VARIABLE.NAME;
    GET(FUSED_WORD,FILE_NAME,FEATURE_VARIABLE.COMPONENTS,
      COMPONENTS_COUNT,TRUE);
    COMPONENTS_COUNT := COMPONENTS_COUNT + FEATURE_VARIABLE.COMPONENTS;
    FEATURE_POINTER := FUSED_FEATURE_LIST.NEXT_LINK(FEATURE_POINTER);
     loop
       FEATURE_VARIABLE := FUSED_FEATURE_LIST.GET_NODE(FEATURE_POINTER);
      FILE_NAME := DIRECTORY_NAME&LIBRARY_PREFIX&WORD_NAME&"."&FEA-
TURE_VARIABLE.NAME;
       GET(FUSED_WORD,FILE_NAME,FEATURE_VARIABLE.COMPONENTS,
    COMPONENTS_COUNT,FALSE);
       COMPONENTS_COUNT := COMPONENTS_COUNT+FEATURE_VARIABLE.COMPONENTS;
       FEATURE_POINTER := FUSED_FEATURE_LIST.NEXT_LINK(FEATURE_POINTER);
     end loop;
  exception
     when FUSED_FEATURE_LIST.UNDER_FLOW =>
       ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
      FILE_NAME := DIRECTORY_NAME&LIBRARY_PREFIX&WORD_NAME&"."&THIS_FEA-
TURE;
      PUT(FUSED_WORD,FILE_NAME);
  end CREATE_FUSION_WORD;


  function TIME_WARP_ALIGNMENT (KNOWN_WORD, UNKNOWN_WORD : in WORD_TYPE)
return WORD_TYPE is
  - [function_header_comment]

     JUMP_UP, JUMP_OVER, JUMP_DIAGONAL, JUMP_UP_DIAGONAL : FLOAT;
     UP_POINTER, RIGHT_POINTER : SLICE_LIST.LIST;
     DISTANCE : FLOAT := 0.0;
     NO_OVER : BOOLEAN := FALSE;
     NEW_WORD : WORD_TYPE;
```

```
 begin
   SLICE_LIST.NULL_POINTER(NEW_WORD.WORD_HEAD);
   SLICE_LIST.NULL_POINTER(NEW_WORD.WORD_TAIL);
   SLICE_LIST.NULL_POINTER(UP_POINTER);
   SLICE_LIST.NULL_POINTER(RIGHT_POINTER);

   CLEAR_LIST(NEW_WORD.WORD_HEAD,NEW_WORD.WORD_TAIL);
   NEW_WORD.WORD_NAME := KNOWN_WORD.WORD_NAME;
   NEW_WORD.WORD_LENGTH := UNKNOWN_WORD.WORD_LENGTH;

   UP_POINTER := KNOWN_WORD.WORD_HEAD;
   RIGHT_POINTER := UNKNOWN_WORD.WORD_HEAD;
   SLICE_LIST.ADD_TO(GET_NODE(UP_POINTER),NEW_WORD.WORD_HEAD,NEW_WORD.
WORD_TAIL);
   RIGHT_POINTER := NEXT_LINK(UNKNOWN_WORD.WORD_HEAD);
    begin
      HORIZONTAL: loop
   if NO_OVER then
      UP_POINTER := NEXT_LINK(UP_POINTER);
      JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
      JUMP_UP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
      if JUMP_DIAGONAL < JUMP_UP_DIAGONAL then
   NULL;
      else
   UP_POINTER := NEXT_LINK(UP_POINTER);
      end if;
      NO_OVER := FALSE;
   else
      JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
      JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
      JUMP_UP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(NEXT_LINK(UP_POINTER))));
      if JUMP_OVER < JUMP_DIAGONAL AND JUMP_OVER < JUMP_UP_DIAGONAL then
   NO_OVER := TRUE;
      elsif JUMP_DIAGONAL < JUMP_OVER AND JUMP_DIAGONAL < JUMP_UP_DIAGONAL
then
   UP_POINTER := NEXT_LINK(UP_POINTER);
      else
   UP_POINTER := NEXT_LINK(NEXT_LINK(UP_POINTER));
```

```
          end if;
        end if;
        SLICE_LIST.ADD_TO(GET_NODE(UP_POINTER),NEW_WORD.WORD_HEAD,NEW_WORD.
WORD_TAIL);
          exit HORIZONTAL when IS_NULL(NEXT_LINK(RIGHT_POINTER));
        RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
          end loop HORIZONTAL;
          return NEW_WORD;
        exception
          when SLICE_LIST.UNDER_FLOW =>
            if IS_NULL(NEXT_LINK(UP_POINTER)) then
              while NOT IS_NULL(RIGHT_POINTER) loop
        SLICE_LIST.ADD_TO(GET_NODE(UP_POINTER),NEW_WORD.WORD_HEAD,NEW_WORD.
WORD_TAIL);
        RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
              end loop;
            else
              GET_UP_LOOP: loop
        JUMP_OVER := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(UP_POINTER));
        JUMP_DIAGONAL := EUCLIDIAN_DIS-
TANCE(GET_NODE(RIGHT_POINTER),GET_NODE(NEXT_LINK(UP_POINTER)));
        if JUMP_OVER < JUMP_DIAGONAL then
          NULL;
        else
          UP_POINTER := NEXT_LINK(UP_POINTER);
        end if;
        SLICE_LIST.ADD_TO(GET_NODE(UP_POINTER),NEW_WORD.WORD_HEAD,NEW_WORD.
WORD_TAIL);
        RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
                exit GET_UP_LOOP when IS_NULL(RIGHT_POINTER) OR
IS_NULL(NEXT_LINK(UP_POINTER));
              end loop GET_UP_LOOP;
              FINSHISH_LOOP: while NOT IS_NULL(RIGHT_POINTER) loop
        while NOT IS_NULL(RIGHT_POINTER) loop

SLICE_LIST.ADD_TO(GET_NODE(UP_POINTER),NEW_WORD.WORD_HEAD,NEW_WORD.WOR
D_TAIL);
          RIGHT_POINTER := NEXT_LINK(RIGHT_POINTER);
        end loop;
              end loop FINSHISH_LOOP;
            end if;
          return NEW_WORD;
        end;
      end TIME_WARP_ALIGNMENT;
```

```
  procedure ERROR_SURFACE (WORD1, WORD2 : in WORD_TYPE; ERROR : out MATRIX_TYPE)
is
  - [procedure_header_comment]
    POINTER1 : SLICE_LIST.list := WORD1.WORD_HEAD;
    POINTER2 : SLICE_LIST.list := WORD2.WORD_HEAD;
    SLICE1, SLICE2 : SLICE_ARRAY_TYPE;
  begin
    for INDEX1 in 1..WORD1.WORD_LENGTH loop
      SLICE1 := GET_NODE(POINTER1);
      POINTER1 := NEXT_LINK(POINTER1);
      for INDEX2 in 1..WORD2.WORD_LENGTH loop
    SLICE2 := GET_NODE(POINTER2);
    POINTER2 := NEXT_LINK(POINTER2);
         ERROR(INDEX1,INDEX2) := EUCLIDIAN_DISTANCE(SLICE1,SLICE2);
      end loop;
      POINTER2 := WORD2.WORD_HEAD;
    end loop;
  exception
    when SLICE_LIST.UNDER_FLOW =>
      PUT_LINE (ITEM => "ERROR IN SLICE COUNTING SOMEWHERE");
  end ERROR_SURFACE;

  procedure PRINT_ERROR_SURFACE (WORD1, WORD2 : in WORD_TYPE; FILE_NAME1,
FILE_NAME2 : in VAR_STRING) is
  - [procedure_header_comment]
    LF_MATRIX : MATRIX_TYPE(0..FILTER-1,0..FILTER-1);
    POINTER1 : SLICE_LIST.list := WORD1.WORD_HEAD;
    POINTER2 : SLICE_LIST.list := WORD2.WORD_HEAD;
    SLICE1, SLICE2 : SLICE_ARRAY_TYPE;
    OUTPUT_FILE : FILE_TYPE;
    ERROR_VALUE : FLOAT;
  begin
    declare
      ERROR_MATRIX : MA-
TRIX_TYPE(1..WORD1.WORD_LENGTH,1..WORD2.WORD_LENGTH);
    begin
      CREATE (
    FILE => OUTPUT_FILE,
    MODE => OUT_FILE,
    NAME => FILE_NAME1);
      PUT (FILE => OUTPUT_FILE, ITEM => WORD2.WORD_LENGTH);
      NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
      for INDEX1 in 1..WORD1.WORD_LENGTH loop
    SLICE1 := GET_NODE(POINTER1);
    POINTER1 := NEXT_LINK(POINTER1);
    for INDEX2 in 1..WORD2.WORD_LENGTH loop
      SLICE2 := GET_NODE(POINTER2);
      POINTER2 := NEXT_LINK(POINTER2);
      ERROR_MATRIX(INDEX1,INDEX2) := EUCLIDIAN_DISTANCE(SLICE1,SLICE2);
      PUT (FILE => OUTPUT_FILE, ITEM => ERROR_MATRIX(INDEX1,INDEX2),
    FORE => 3,
    AFT => 5,
    EXP => 0);
    end loop;
    NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
    POINTER2 := WORD2.WORD_HEAD;
      end loop;
      CLOSE (FILE => OUTPUT_FILE);
      TWO_DEE_FFT(ERROR_MA-
TRIX,WORD1.WORD_LENGTH,WORD2.WORD_LENGTH,LF_MATRIX );
      CREATE (
        FILE => OUTPUT_FILE,
        MODE => OUT_FILE,
        NAME => FILE_NAME2);
      for INDEX1 in 0..FILTER-1 loop
        for INDEX2 in 0..FILTER-1 loop
          PUT (FILE => OUTPUT_FILE, ITEM => LF_MATRIX(INDEX1,INDEX2),
            FORE => 8,
            AFT => 8,
            EXP => 0);
        end loop;
        NEW_LINE (FILE => OUTPUT_FILE, SPACING => 1);
      end loop;
      CLOSE (FILE => OUTPUT_FILE);
    end;
  end PRINT_ERROR_SURFACE;

  procedure OUTPUT_ERROR_SURFACE (WORD_NAME1, WORD_NAME2, FILE_NAME : in
VAR_STRING) is
  - [procedure_header_comment]
    WORD1, WORD2, UNKNOWN_WORD : WORD_TYPE;
    THIS_FEATURE, WORD_NAME : VAR_STRING;
  begin
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_TAIL);
    SLICE_LIST.NULL_POINTER(WORD1.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(WORD1.WORD_TAIL);
    SLICE_LIST.NULL_POINTER(WORD2.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(WORD2.WORD_TAIL);
    ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
    WORD_NAME := DIRECTORY_NAME&WORD_NAME1&"."&THIS_FEATURE;
    GET(WORD1,WORD_NAME);
    WORD_NAME := DIRECTORY_NAME&WORD_NAME2&"."&THIS_FEATURE;
    GET(WORD2,WORD_NAME);
    PRINT_ERROR_SURFACE(WORD1,WORD2,"ER_"&FILE_NAME&"."&THIS_FEA-
TURE,"LF_"&FILE_NAME&"."&THIS_FEATURE);
  end OUTPUT_ERROR_SURFACE;

  function RECOGNIZE_ERROR (NAME : in VAR_STRING) return RESULT_RECORD_TYPE is
  - [function_header_comment]

    TEST_LF_MATRIX, LF_MATRIX : MATRIX_TYPE(0..FILTER-1,0..FILTER-1);
    UNKNOWN_WORD, KNOWN_WORD : WORD_TYPE;
    WORD_NAME, THIS_FEATURE : VAR_STRING;
    WORD_POINTER : WORD_LIST.LIST := LIBRARY_HEAD;
    WORD_NAME_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
    ER_RESULTS : RESULT_RECORD_TYPE;
    DISTANCE, NORMAL : FLOAT := 0.0;
    WORD_COUNT, SMALLEST : INTEGER := 0;
  begin
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(UNKNOWN_WORD.WORD_TAIL);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_HEAD);
    SLICE_LIST.NULL_POINTER(KNOWN_WORD.WORD_TAIL);
    RESULT_LIST.NULL_POINTER(ER_RESULTS.RESULT_HEAD);
    RESULT_LIST.NULL_POINTER(ER_RESULTS.RESULT_TAIL);
    ASSIGN(THIS_FEATURE,NAME_OF_FEATURE);
    WORD_NAME := DIRECTORY_NAME&NAME&"."&THIS_FEATURE;
    GET(UNKNOWN_WORD,WORD_NAME);
    declare
      ERROR_MATRIX : MATRIX_TYPE(1..unknown_WORD.WORD_LENGTH,1..UN-
KNOWN_WORD.WORD_LENGTH);
    begin
      ERROR_SURFACE(UNKNOWN_WORD,UNKNOWN_WORD,ERROR_MATRIX);
      TWO_DEE_FFT(ERROR_MATRIX,UNKNOWN_WORD.WORD_LENGTH,UN-
KNOWN_WORD.WORD_LENGTH,TEST_LF_MATRIX );
    end;
    begin
    loop
   NEXT_WORD(WORD_NAME,WORD_NAME_POINTER);
   WORD_COUNT := WORD_COUNT + 1;
   GET_WORD: loop
     KNOWN_WORD := GET_NODE(WORD_POINTER);
     WORD_POINTER := NEXT_LINK(WORD_POINTER);
   exit GET_WORD when IS_EQUAL(KNOWN_WORD.WORD_NAME,WORD_NAME);
   end loop GET_WORD;
   KNOWN_WORD := TIME_WARP_ALIGNMENT(KNOWN_WORD,UNKNOWN_WORD);
   declare
     ERROR_MATRIX : MATRIX_TYPE(1..KNOWN_WORD.WORD_LENGTH,1..UN-
KNOWN_WORD.WORD_LENGTH);
   begin
     ERROR_SURFACE(KNOWN_WORD,UNKNOWN_WORD,ERROR_MATRIX);
     TWO_DEE_FFT(ERROR_MATRIX,KNOWN_WORD.WORD_LENGTH,UN-
KNOWN_WORD.WORD_LENGTH,LF_MATRIX );
   end;
   DISTANCE := TWO_DEE_EUCLIDIAN(TEST_LF_MATRIX,LF_MATRIX);
   RESULT_LIST.ADD_TO(DISTANCE,ER_RESULTS.RESULT_HEAD,ER_RESULTS.RE-
SULT_TAIL);
   NORMAL := NORMAL + DISTANCE**2;
     end loop;
    exception
      when DATABASE_LIST.UNDER_FLOW =>
        NULL;
    end;
   NORMAL := SQRT(NORMAL);
   for INDEX in 1..WORD_COUNT loop
     RESULT_LIST.REMOVE_NEXT_NODE(DISTANCE,ER_RESULTS.RESULT_HEAD,ER_RE-
SULTS.RESULT_TAIL);
     DISTANCE := DISTANCE/NORMAL;
     RESULT_LIST.ADD_TO(DISTANCE,ER_RESULTS.RESULT_HEAD,ER_RESULTS.RE-
SULT_TAIL);
   end loop;
    return ER_RESULTS;
  end RECOGNIZE_ERROR;


  function "+" (LEFT, RIGHT : in SLICE_ARRAY_TYPE) return SLICE_ARRAY_TYPE is
  - [function_header_comment]
    NEW_SLICE : SLICE_ARRAY_TYPE;
  begin
    for INDEX in 1..NUMBER_OF_COMPONENTS loop
      NEW_SLICE(INDEX) := LEFT(INDEX) + RIGHT(INDEX);
    end loop;
    return NEW_SLICE;
  end "+";

  function "+" (LEFT, RIGHT : in WORD_TYPE) return WORD_TYPE is
  - [function_header_comment]
    NEW_WORD : WORD_TYPE;
    RIGHT_SLICE_POINTER, LEFT_SLICE_POINTER : SLICE_LIST.LIST;
  begin
    RIGHT_SLICE_POINTER := RIGHT.WORD_HEAD;
    LEFT_SLICE_POINTER := LEFT.WORD_HEAD;
    for INDEX in 1..LEFT.WORD_LENGTH loop

SLICE_LIST.ADD_TO(GET_NODE(LEFT_SLICE_POINTER)+GET_NODE(RIGHT_SLICE_POINTER
),NEW_WORD.WORD_HEAD,NEW_WORD.WORD_TAIL);
      RIGHT_SLICE_POINTER := NEXT_LINK(RIGHT_SLICE_POINTER);
      LEFT_SLICE_POINTER := NEXT_LINK(LEFT_SLICE_POINTER);
    end loop;
    NEW_WORD.WORD_LENGTH := LEFT.WORD_LENGTH;
    return NEW_WORD;
  end "+";

  function "/" (LEFT : in SLICE_ARRAY_TYPE; RIGHT : in FLOAT) return SLICE_ARRAY_TYPE is
  - [function_header_comment]
    NEW_SLICE : SLICE_ARRAY_TYPE;
  begin
    for INDEX in 1..NUMBER_OF_COMPONENTS loop
      NEW_SLICE(INDEX) := LEFT(INDEX) / RIGHT;
    end loop;
    return NEW_SLICE;
  end "/";

  function "/" (LEFT : in WORD_TYPE; RIGHT : in FLOAT ) return WORD_TYPE is
  - [function_header_comment]
    NEW_WORD : WORD_TYPE;
    LEFT_SLICE_POINTER : SLICE_LIST.LIST;
  begin
    LEFT_SLICE_POINTER := LEFT.WORD_HEAD;
    for INDEX in 1..LEFT.WORD_LENGTH loop

SLICE_LIST.ADD_TO(GET_NODE(LEFT_SLICE_POINTER)/RIGHT,NEW_WORD.WORD_HEAD,N
EW_WORD.WORD_TAIL);
      LEFT_SLICE_POINTER := NEXT_LINK(LEFT_SLICE_POINTER);
    end loop;
    NEW_WORD.WORD_LENGTH := LEFT.WORD_LENGTH;
    return NEW_WORD;
  end "/";

  function "*" (LEFT : in SLICE_ARRAY_TYPE; RIGHT : in FLOAT) return SLICE_ARRAY_TYPE is
  - [function_header_comment]
    NEW_SLICE : SLICE_ARRAY_TYPE;
  begin
    for INDEX in 1..NUMBER_OF_COMPONENTS loop
      NEW_SLICE(INDEX) := LEFT(INDEX) * RIGHT;
    end loop;
    return NEW_SLICE;
  end "*";

  function "*" (LEFT : in WORD_TYPE; RIGHT : in FLOAT ) return WORD_TYPE is
  - [function_header_comment]
    NEW_WORD : WORD_TYPE;
    LEFT_SLICE_POINTER : SLICE_LIST.LIST;
  begin
    LEFT_SLICE_POINTER := LEFT.WORD_HEAD;
    for INDEX in 1..LEFT.WORD_LENGTH loop

SLICE_LIST.ADD_TO(GET_NODE(LEFT_SLICE_POINTER)*RIGHT,NEW_WORD.WORD_HEAD,N
EW_WORD.WORD_TAIL);
      LEFT_SLICE_POINTER := NEXT_LINK(LEFT_SLICE_POINTER);
    end loop;
    NEW_WORD.WORD_LENGTH := LEFT.WORD_LENGTH;
    return NEW_WORD;
  end "*";

  procedure COPY_WORDS (WORD1, WORD2 : in out WORD_TYPE) is
  - [procedure_header_comment]
  begin
    WORD2.WORD_LENGTH := WORD1.WORD_LENGTH;
    SLICE_LIST.COPY_LIST(WORD1.WORD_HEAD,WORD1.WORD_TAIL,WORD2.WORD_HEAD
,WORD2.WORD_TAIL);
  end COPY_WORDS;

  procedure CREATE_TW_TEMPLATE (PREFIX_OUT, WORD_NAME : in VAR_STRING) is
  - [procedure_header_comment]

    COUNT : INTEGER := 1;
    INPUT_FILE, OUTPUT_FILE : FILE_TYPE;
    THIS_PREFIX, FILENAME : VAR_STRING;
    PREFIX_POINTER : PREFIX_LIST.LIST := PREFIX_HEAD;
    TEMPLATE_WORD, LONGEST_WORD, OTHER_WORD : WORD_TYPE;

    TEMPLATE_POINTER, TEMPLATE_HEAD, TEMPLATE_TAIL : WORD_LIST.LIST;

  begin
    begin
      THIS_PREFIX := PREFIX_LIST.GET_NODE(PREFIX_POINTER);
      PREFIX_POINTER := PREFIX_LIST.NEXT_LINK(PREFIX_POINTER);
      FILENAME := DIRECTORY_NAME&THIS_PREFIX&WORD_NAME&"."&NAME_OF_FEA-
TURE;
      GET(LONGEST_WORD, FILENAME);
      loop
    THIS_PREFIX := PREFIX_LIST.GET_NODE(PREFIX_POINTER);
    PREFIX_POINTER := PREFIX_LIST.NEXT_LINK(PREFIX_POINTER);
    COUNT := COUNT + 1;
    FILENAME := DIRECTORY_NAME&THIS_PREFIX&WORD_NAME&"."&NAME_OF_FEA-
TURE;
    GET(OTHER_WORD, FILENAME);
        if OTHER_WORD.WORD_LENGTH > LONGEST_WORD.WORD_LENGTH then
          WORD_LIST.ADD_TO(LONGEST_WORD,TEMPLATE_HEAD,TEMPLATE_TAIL);
      COPY_WORDS(OTHER_WORD,LONGEST_WORD);
        else
          WORD_LIST.ADD_TO(OTHER_WORD,TEMPLATE_HEAD,TEMPLATE_TAIL);
        end if;
      end loop;
    exception
      when PREFIX_LIST.UNDER_FLOW =>
        TEMPLATE_POINTER := TEMPLATE_HEAD;
    COPY_WORDS(LONGEST_WORD,TEMPLATE_WORD);
    end;
    begin
      loop
        OTHER_WORD := GET_NODE(TEMPLATE_POINTER);
    TEMPLATE_POINTER := NEXT_LINK(TEMPLATE_POINTER);
    OTHER_WORD := TIME_WARP_ALIGNMENT(OTHER_WORD,LONGEST_WORD);
    TEMPLATE_WORD := TEMPLATE_WORD + OTHER_WORD;
      end loop;
    exception
      when WORD_LIST.UNDER_FLOW =>
    TEMPLATE_WORD := TEMPLATE_WORD / FLOAT(COUNT);
```

```
        FILENAME := DIRECTORY_NAME&PRE-
FIX_OUT&WORD_NAME&"."&NAME_OF_FEATURE;
    end;
    PUT(TEMPLATE_WORD, FILENAME);
  end CREATE_TW_TEMPLATE;


  procedure UPDATE_TW_TEMPLATE (PREFIX_IN, PREFIX_OUT, WORD_NAME : in
VAR_STRING;
      TEMPLATE_FACTOR : in FLOAT) is
  - [procedure_header_comment]

    COUNT : INTEGER := 1;
    INPUT_FILE, OUTPUT_FILE : FILE_TYPE;
    THIS_PREFIX, FILENAME : VAR_STRING;
    PREFIX_POINTER : PREFIX_LIST.LIST := PREFIX_HEAD;
    TEMPLATE_WORD, LONGEST_WORD, OTHER_WORD : WORD_TYPE;

    TEMPLATE_POINTER, TEMPLATE_HEAD, TEMPLATE_TAIL : WORD_LIST.LIST;

  begin
    FILENAME := DIRECTORY_NAME&PREFIX_IN&WORD_NAME&"."&NAME_OF_FEATURE,
    GET(OTHER_WORD, FILENAME);
    FILENAME := DIRECTORY_NAME&PREFIX_OUT&WORD_NAME&"."&NAME_OF_FEA-
TURE;
    GET(LONGEST_WORD, FILENAME);
    OTHER_WORD := TIME_WARP_ALIGNMENT(OTHER_WORD,LONGEST_WORD);
    TEMPLATE_WORD := (LONGEST_WORD * TEMPLATE_FACTOR +
OTHER_WORD)/(1.0+TEMPLATE_FACTOR);
    PUT(TEMPLATE_WORD, FILENAME);
  end UPDATE_TW_TEMPLATE;

end REC_SUPPORT_PACKAGE;
```

```
with TEXT_IO, DATABASE_PACKAGE, STRING_OPERATIONS, LINKED_LIST, FU-
SION_REC_PACKAGE;
use TEXT_IO, DATABASE_PACKAGE, STRING_OPERATIONS, FUSION_REC_PACKAGE;
package FUSION_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    THIS PACKAGE CONTAINS THE FUNCTION AND PROCEDURES NECESSARY FOR FUSING
-  AND DISPLAYING THE RESULTS FROM MULTIPLE FEATURE AND MULTIPLE RECOG-
NIZERS
-
- [optional package tags]
- -
  use FUSION_LIST;

  THE_FILE, DUMMY : VAR_STRING;
  REPORT_FILE : FILE_TYPE;
  OUTPUT_CHANNEL : OUTPUT_CHANNEL_TYPE;

  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    This procedure outputs the recognition results for each word for
  -  a speified feature
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure FEATURE_RESULTS_OUTPUT (OUTPUT_CHANNEL : in OUTPUT_CHANNEL_TYPE;
        REPORT_FILE : in out FILE_TYPE);
  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    This procedure ouputs the recognition results for each word for
  -  all features
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
```

```
 - -
 procedure ALL_FEATURE_RESULTS_OUTPUT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
          REPORT_FILE : in out FILE_TYPE);
 - ++
 -
 - FUNCTIONAL DESCRIPTION:
 -
 -    This procedure outputs the name of the word with the smallest
 -  recognition distance for a specific feature
 -
 - [formal parameters]
 - [design]
 - [optional subprogram tags]
 - -
 procedure FEATURE_SMALLEST_RESULT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
        REPORT_FILE : in out FILE_TYPE);
 - ++
 -
 - FUNCTIONAL DESCRIPTION:
 -
 -    The procedure outputs the name of the word with the smallest
 -
 -
 - [formal parameters]
 - [design]
 - [optional subprogram tags]
 - -
 procedure ALL_FEATURE_SMALLEST_RESULT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
          REPORT_FILE : in out FILE_TYPE);
 - [procedure_header_comment]
 procedure FUSION;

 - [procedure_header_comment]
 procedure FUSION_RESULT;

 - [procedure_header_comment]
 procedure FUSION_CONTROL_PANEL;

 - [procedure_header_comment]
 procedure CREATE_SUBLIST (NUMBER_WORDS : in INTEGER);
```

```
  - [function_header_comment]
  function AGREE return FLOAT;

  package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
  package INTEGER_IO is nt w TEXT_IO.INTEGER_IO (INTEGER);
-   package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (FEATURE_NAME_TYPE);
  use FLOAT_IO, INTEGER_IO;
end FUSION_PACKAGE;
```

```
with FLOAT_MATH_LIB;
use FLOAT_MATH_LIB;
package body FUSION_PACKAGE is
- [package_body_header_comment]

  procedure MATCH_FEATURE (ITEM_ONE, ITEM_TWO : in out RESULT_RECORD_TYPE; CON-
TINUE : out BOOLEAN) is
  - [procedure_header_comment]
  begin
    if ITEM_ONE.FEATURE_NAME = ITEM_TWO.FEATURE_NAME then
      CONTINUE := FALSE;
      ITEM_TWO := ITEM_ONE;
    else
      CONTINUE := TRUE;
    end if;
  end MATCH_FEATURE;

  procedure GET_FEATURE_NODE is new FUSION_LIST.ITERATE_LIST (PROCESS =>
MATCH_FEATURE);

  procedure FEATURE_RESULTS_OUTPUT (OUTPUT_CHANNEL : in OUTPUT_CHANNEL_TYPE;
       REPORT_FILE : in out FILE_TYPE) is
  - [procedure_header_comment]
    RESULTS : RESULT_RECORD_TYPE;
    WORD : VAR_STRING;
    WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
    RESULTS_POINTER : RESULT_LIST.LIST;
    DISTANCE : FLOAT;

  begin
    RESULT_LIST.NULL_POINTER(RESULTS_POINTER);
    NEW_LINE;
    PUT_LINE (ITEM => "TYPE IN NAME OF FEATURE");
    GET_LINE (ITEM => RESULTS.FEATURE_NAME);
    GET_FEATURE_NODE(RESULTS,FUSION_HEAD,FUSION_TAIL);
    RESULTS_POINTER := RESULTS.RESULT_HEAD;
    if OUTPUT_CHANNEL = SCREEN then
      NEW_LINE;
      PUT (ITEM => "RESULTS FOR FEATURE,  ");
      PUT (ITEM => RESULTS.FEATURE_NAME);
      NEW_LINE;
    else
      NEW_LINE (FILE => REPORT_FILE);
      PUT (FILE => REPORT_FILE, ITEM => "RESULTS FOR FEATURE,  ");
      PUT (FILE => REPORT_FILE, ITEM => RESULTS.FEATURE_NAME);
      NEW_LINE (FILE => REPORT_FILE);
    end if;
    begin
      loop
    DISTANCE := RESULT_LIST.GET_NODE(RESULTS_POINTER);
    RESULTS_POINTER := RESULT_LIST.NEXT_LINK(RESULTS_POINTER);
    NEXT_WORD(WORD,WORD_POINTER);
    if OUTPUT_CHANNEL = SCREEN then
      PUT (ITEM => WORD);
      for INDEX1 in 1..20-STRING_LENGTH(WORD) loop
    PUT (ITEM => ' ');
      end loop;
      FLOAT_IO.PUT (ITEM => DISTANCE,
    FORE => 1,
    AFT => 4,
    EXP => 0);
      NEW_LINE;
    else
      PUT (FILE => REPORT_FILE, ITEM => WORD);
      for INDEX1 in 1..20-STRING_LENGTH(WORD) loop
    PUT (FILE => REPORT_FILE, ITEM => ' ');
      end loop;
      FLOAT_IO.PUT (FILE => REPORT_FILE, ITEM => DISTANCE,
    FORE => 1,
    AFT => 4,
    EXP => 0);
      NEW_LINE (FILE => REPORT_FILE);
    end if;
      end loop;
     exception
       when RESULT_LIST.UNDER_FLOW =>
         NULL;
     end;
  exception
    when FUSION_LIST.NOT_IN_LIST =>
      PUT_LINE (ITEM => "THAT FEATURE HAS NO RECOGNITION RESULTS");
  end FEATURE_RESULTS_OUTPUT;

  procedure ALL_FEATURE_RESULTS_OUTPUT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
```

```
        REPORT_FILE : in out FILE_TYPE) is
- [procedure_header_comment]
  DISTANCE : FLOAT := 0.0;
  WORD : VAR_STRING;
  RESULTS : RESULT_RECORD_TYPE;
  RESULTS_POINTER : FUSION_LIST.LIST := FUSION_HEAD;
  WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
  DISTANCE_POINTER : RESULT_LIST.LIST;
  COUNTER : INTEGER := 0;
begin
  RESULT_LIST.NULL_POINTER(DISTANCE_POINTER);

  if OUTPUT_CHANNEL = SCREEN then
    PUT (ITEM => "RESULTS FOR FEATURE,  ");
  else
    PUT (FILE => REPORT_FILE, ITEM => "RESULTS FOR FEATURE,  ");
  end if;
  while NOT IS_NULL(RESULTS_POINTER) loop
    RESULTS := GET_NODE(RESULTS_POINTER);
    if OUTPUT_CHANNEL = SCREEN then
  PUT (ITEM => RESULTS.FEATURE_NAME);
  PUT (ITEM => " ");
    else
  PUT (FILE => REPORT_FILE, ITEM => RESULTS.FEATURE_NAME);
  PUT (FILE => REPORT_FILE, ITEM => " ");
    end if;
    RESULTS_POINTER := NEXT_LINK(RESULTS_POINTER);
  end loop;
  if OUTPUT_CHANNEL = SCREEN then
    NEW_LINE;
  else
    NEW_LINE (FILE => REPORT_FILE);
  end if;
   begin
    loop
  NEXT_WORD(WORD,WORD_POINTER);
  COUNTER := COUNTER + 1;
  RESULTS_POINTER := FUSION_HEAD;
       if OUTPUT_CHANNEL = SCREEN then
    PUT (ITEM => WORD);
    for INDEX1 in 1..20-STRING_LENGTH(WORD) loop
  PUT (ITEM => ' ');
```

```
    end loop;
      else
    PUT (FILE => REPORT_FILE, ITEM => WORD);
    for INDEX1 in 1..20-STRING_LENGTH(WORD) loop
  PUT (FILE => REPORT_FILE, ITEM => ' ');
    end loop;
      end if;
      begin
        loop
          RESULTS := GET_NODE(RESULTS_POINTER);
  RESULTS_POINTER := NEXT_LINK(RESULTS_POINTER);
  DISTANCE_POINTER := RESULTS.RESULT_HEAD;
          for INDEX in 1..COUNTER loop
            DISTANCE := RESULT_LIST.GET_NODE(DISTANCE_POINTER);
    DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
          end loop;
          if OUTPUT_CHANNEL = SCREEN then
            PUT (ITEM => DISTANCE,
              FORE => 2,
              AFT => 4,
              EXP => 0);
          else
            PUT (FILE => REPORT_FILE, ITEM => DISTANCE,
              FORE => 2,
              AFT => 4,
              EXP => 0);
          end if;
        end loop;
      exception
        when FUSION_LIST.UNDER_FLOW =>
          if OUTPUT_CHANNEL = SCREEN then
            NEW_LINE (SPACING => 1);
          else
            NEW_LINE (FILE => REPORT_FILE, SPACING => 1);
          end if;
      end;
    end loop;
  exception
    when DATABASE_LIST.UNDER_FLOW =>
      NULL;
  end;
end ALL_FEATURE_RESULTS_OUTPUT;
```

```
function SMALL_VALUE (RESULT : in RESULT_RECORD_TYPE) return RESULT_LIST.LIST is
- [function_header_comment]
  DISTANCE_POINTER, SMALLEST : RESULT_LIST.LIST := RESULT.RESULT_HEAD;
begin
  loop
    DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
    if RESULT_LIST.GET_NODE(DISTANCE_POINTER) < RESULT_LIST.GET_NODE(SMALL-
EST) then
  SMALLEST := DISTANCE_POINTER;
    end if;
  end loop;
exception
  when RESULT_LIST.UNDER_FLOW =>
    return SMALLEST;
end SMALL_VALUE;

function SMALL_VALUE_BY_INDEX (RESULT : in RESULT_RECORD_TYPE) return INTEGER is
- [function_header_comment]
  COUNTER : INTEGER := 1;
  SMALL_COUNT : INTEGER := 1;
  DISTANCE_POINTER, SMALLEST : RESULT_LIST.LIST := RESULT.RESULT_HEAD;
begin
  loop
    DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
    COUNTER := COUNTER + 1;
    if RESULT_LIST.GET_NODE(DISTANCE_POINTER) < RESULT_LIST.GET_NODE(SMALL-
EST) then
  SMALLEST := DISTANCE_POINTER;
  SMALL_COUNT := COUNTER;
    end if;
  end loop;
exception
  when RESULT_LIST.UNDER_FLOW =>
    return SMALL_COUNT;
end SMALL_VALUE_BY_INDEX;

procedure FEATURE_SMALLEST_RESULT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
        REPORT_FILE : in out FILE_TYPE) is
- [procedure_header_comment]
  RESULTS : RESULT_RECORD_TYPE;
  SMALLEST : RESULT_LIST.LIST;
  WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
  WORD : VAR_STRING;
  DISTANCE_POINTER : RESULT_LIST.LIST;
  DISTANCE : FLOAT;
begin
  RESULT_LIST.NULL_POINTER(DISTANCE_POINTER);
  RESULT_LIST.NULL_POINTER(SMALLEST);
  NEW_LINE;
  PUT_LINE (ITEM => "TYPE IN NAME OF FEATURE");
  GET_LINE (ITEM => RESULTS.FEATURE_NAME);
  GET_FEATURE_NODE(RESULTS,FUSION_HEAD,FUSION_TAIL);
  if OUTPUT_CHANNEL = SCREEN then
    PUT (ITEM => "RESULTS FOR FEATURE, ");
    PUT (ITEM => RESULTS.FEATURE_NAME);
    NEW_LINE;
  else
    PUT (FILE => REPORT_FILE, ITEM => "RESULTS FOR FEATURE, ");
    PUT (FILE => REPORT_FILE, ITEM => RESULTS.FEATURE_NAME);
    NEW_LINE (FILE => REPORT_FILE);
  end if;
  SMALLEST := SMALL_VALUE(RESULTS);
  DISTANCE := RESULT_LIST.GET_NODE(SMALLEST);
  DISTANCE_POINTER := RESULTS.RESULT_HEAD;
   loop
     exit when RESULT_LIST.IS_EQUAL(SMALLEST,DISTANCE_POINTER);
     WORD_POINTER := DATABASE_LIST.NEXT_LINK(WORD_POINTER);
     DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
   end loop;
  WORD := DATABASE_LIST.GET_NODE(WORD_POINTER);
  if OUTPUT_CHANNEL = SCREEN then
    PUT (ITEM => "WORD - ");
    PUT (ITEM => WORD);
    PUT (ITEM => " HAS THE SMALLEST DTW DISTANCE OF ");
    FLOAT_IO.PUT (ITEM => DISTANCE,
  FORE => 2,
  AFT => 4,
  EXP => 0);
    NEW_LINE;
  else
    PUT (FILE => REPORT_FILE, ITEM => "WORD - ");
    PUT (FILE => REPORT_FILE, ITEM => WORD);
    PUT (FILE => REPORT_FILE, ITEM => " HAS THE SMALLEST DTW DISTANCE OF ");
```

```
      FLOAT_IO.PUT (FILE => REPORT_FILE, ITEM => DISTANCE,
    FORE => 2,
    AFT => 4,
    EXP => 0);
      NEW_LINE (FILE => REPORT_FILE);
    end if;
  end FEATURE_SMALLEST_RESULT;

  procedure ALL_FEATURE_SMALLEST_RESULT (OUTPUT_CHANNEL : in OUTPUT_CHAN-
NEL_TYPE;
          REPORT_FILE : in out FILE_TYPE) is
  - [procedure_header_comment]
    RESULTS : RESULT_RECORD_TYPE;
    WORD : VAR_STRING;
    RESULTS_POINTER : FUSION_LIST.LIST := FUSION_HEAD;
    WORD_POINTER : DATABASE_LIST.LIST;
    SMALLEST, DISTANCE_POINTER : RESULT_LIST.LIST;
    DISTANCE : FLOAT;
  begin
    DATABASE_LIST.NULL_POINTER(WORD_POINTER);
    while NOT IS_NULL(RESULTS_POINTER) loop
      RESULTS := GET_NODE(RESULTS_POINTER);
      if OUTPUT_CHANNEL = SCREEN then
    PUT (ITEM => "RESULTS FOR FEATURE, ");
    PUT (ITEM => RESULTS.FEATURE_NAME);
    NEW_LINE;
      else
    PUT (FILE => REPORT_FILE, ITEM => "RESULTS FOR FEATURE, ");
    PUT (FILE => REPORT_FILE, ITEM => RESULTS.FEATURE_NAME);
    NEW_LINE (FILE => REPORT_FILE);
      end if;
      SMALLEST := SMALL_VALUE(RESULTS);
      WORD_POINTER := DATABASE_HEAD;
      DISTANCE := RESULT_LIST.GET_NODE(SMALLEST);
      DISTANCE_POINTER := RESULTS.RESULT_HEAD;
      loop
    exit when RESULT_LIST.IS_EQUAL(SMALLEST,DISTANCE_POINTER);
    WORD_POINTER := DATABASE_LIST.NEXT_LINK(WORD_POINTER);
    DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
      end loop;
      WORD := DATABASE_LIST.GET_NODE(WORD_POINTER);
      if OUTPUT_CHANNEL = SCREEN then
  PUT (ITEM => "WORD - ");
  PUT (ITEM => WORD);
  PUT (ITEM => " HAS THE SMALLEST DTW DISTANCE OF ");
  FLOAT_IO.PUT (ITEM => DISTANCE,
    FORE => 2,
    AFT => 4,
    EXP => 0);
  NEW_LINE;
    else
  PUT (FILE => REPORT_FILE, ITEM => "WORD - ");
  PUT (FILE => REPORT_FILE, ITEM => WORD);
  PUT (FILE => REPORT_FILE, ITEM => " HAS THE SMALLEST DTW DISTANCE OF ");
  FLOAT_IO.PUT (FILE => REPORT_FILE, ITEM => DISTANCE,
    FORE => 2,
    AFT => 4,
    EXP => 0);
  NEW_LINE (FILE => REPORT_FILE);
    end if;
    RESULTS_POINTER := NEXT_LINK(RESULTS_POINTER);
  end loop;
end ALL_FEATURE_SMALLEST_RESULT;

procedure FUSION is
- [procedure_header_comment]
  RESULTS : RESULT_RECORD_TYPE;
  FUSED_RESULTS : RESULT_RECORD_TYPE;
  RESULTS_POINTER : FUSION_LIST.LIST := FUSION_HEAD;
  DISTANCE_POINTER : RESULT_LIST.LIST;
  FUSED_DISTANCE, DISTANCE : FLOAT;
begin
  RESULT_LIST.NULL_POINTER(DISTANCE_POINTER);
  FUSED_RESULTS.FUSION_FACTOR := 1.0;
  ASSIGN(FUSED_RESULTS.FEATURE_NAME,"FUS");
  RESULTS := GET_NODE(RESULTS_POINTER);
  DISTANCE_POINTER := RESULTS.RESULT_HEAD;
  begin
    loop
      DISTANCE := RESULT_LIST.GET_NODE(DISTANCE_POINTER);
  DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
  DISTANCE := DISTANCE * RESULTS.FUSION_FACTOR;
  RESULT_LIST.ADD_TO(DISTANCE,FUSED_RESULTS.RESULT_HEAD,FUSED_RESULTS.RE-
SULT_TAIL);
```

```
        end loop;
      exception
        when RESULT_LIST.UNDER_FLOW =>
          NULL;
      end;
      RESULTS_POINTER := NEXT_LINK(RESULTS_POINTER);
      while NOT IS_NULL(RESULTS_POINTER) loop
        RESULTS := GET_NODE(RESULTS_POINTER);
        begin
      DISTANCE_POINTER := RESULTS.RESULT_HEAD;
          loop
            DISTANCE := RESULT_LIST.GET_NODE(DISTANCE_POINTER);
        RESULT_LIST.REMOVE_NEXT_NODE(FUSED_DISTANCE,FUSED_RESULTS.RE-
SULT_HEAD,FUSED_RESULTS.RESULT_TAIL);
        FUSED_DISTANCE := FUSED_DISTANCE + RESULTS.FUSION_FACTOR*DISTANCE;
        RESULT_LIST.ADD_TO(FUSED_DISTANCE,FUSED_RESULTS.RESULT_HEAD,FUSED_RE-
SULTS.RESULT_TAIL);
        DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
          end loop;
        exception
          when RESULT_LIST.UNDER_FLOW =>
            NULL;
        end;
        RESULTS_POINTER := NEXT_LINK(RESULTS_POINTER);
      end loop;
      ADD_TO(FUSED_RESULTS,FUSION_HEAD,FUSION_TAIL);
    end FUSION;

    procedure FUSION_RESULT is
    - [procedure_header_comment]
      RESULTS : RESULT_RECORD_TYPE;
      DISTANCE_POINTER, SMALLEST : RESULT_LIST.LIST;
      WORD_POINTER : DATABASE_LIST.LIST := DATABASE_HEAD;
      WORD : VAR_STRING;
    begin
      ASSIGN(RESULTS.FEATURE_NAME,"FUS");
      GET_FEATURE_NODE(RESULTS,FUSION_HEAD,FUSION_TAIL);
      PUT (ITEM => "FUSION RESULTS IS ");
      SMALLEST := SMALL_VALUE(RESULTS);
      DISTANCE_POINTER := RESULTS.RESULT_HEAD;
      loop
        exit when RESULT_LIST.IS_EQUAL(SMALLEST,DISTANCE_POINTER);
        WORD_POINTER := DATABASE_LIST.NEXT_LINK(WORD_POINTER);
    DISTANCE_POINTER := RESULT_LIST.NEXT_LINK(DISTANCE_POINTER);
  end loop;
  WORD := DATABASE_LIST.GET_NODE(WORD_POINTER);
  PUT (ITEM => WORD);
  NEW_LINE;
end FUSION_RESULT;

procedure CREATE_SUBLIST (NUMBER_WORDS : in INTEGER) is
- [procedure_header_comment]
  package TEMP_LIST is new LINKED_LIST (ITEM_TYPE => FLOAT);
  use TEMP_LIST;
  TEMP_HEAD, TEMP_TAIL : TEMP_LIST.LIST;
  RESULTS : RESULT_RECORD_TYPE;
  RESULTS_POINTER : RESULT_LIST.LIST;
  DISTANCE, SMALL : FLOAT;
  TEMP_POINTER, SMALLEST : TEMP_LIST.LIST;
  SMALLEST_IN_SUBLIST, SUBLIST_POINTER : DATABASE_LIST.LIST := SUBLIST_HEAD;
  WORD : VAR_STRING;
  WORD_POINTER : DATABASE_LIST.LIST := LIST_HEAD;
begin
  NULL_POINTER(TEMP_POINTER);
  NULL_POINTER(SMALLEST);
  DATABASE_LIST.CLEAR_LIST(SUBLIST_HEAD,SUBLIST_TAIL);
   ASSIGN(RESULTS.FEATURE_NAME,"FUS");
  GET_FEATURE_NODE(RESULTS,FUSION_HEAD,FUSION_TAIL);
  RESULTS_POINTER := RESULTS.RESULT_HEAD;
  for INDEX in 1..NUMBER_WORDS loop
     DISTANCE := RESULT_LIST.GET_NODE(RESULTS_POINTER);
    RESULTS_POINTER := RESULT_LIST.NEXT_LINK(RESULTS_POINTER);
    TEMP_LIST.ADD_TO(DISTANCE,TEMP_HEAD,TEMP_TAIL);
    NEXT_WORD(WORD,WORD_POINTER);
     DATABASE_LIST.ADD_TO(WORD,SUBLIST_HEAD,SUBLIST_TAIL);
  end loop;
  begin
     loop
  begin
    SMALLEST := TEMP_HEAD;
    TEMP_POINTER := TEMP_HEAD;
    SMALLEST_IN_SUBLIST := SUBLIST_HEAD;
    SUBLIST_POINTER := SUBLIST_HEAD;
    loop
  TEMP_POINTER := TEMP_LIST.NEXT_LINK(TEMP_POINTER);
```

```
    SUBLIST_POINTER := DATABASE_LIST.NEXT_LINK(SUBLIST_POINTER);
    if TEMP_LIST.GET_NODE(TEMP_POINTER) > TEMP_LIST.GET_NODE(SMALLEST) then
      SMALLEST := TEMP_POINTER;
      SMALLEST_IN_SUBLIST := SUBLIST_POINTER;
    end if;
      end loop;
    exception
      when TEMP_LIST.UNDER_FLOW =>
    NULL;
    end;
    DISTANCE := TEMP_LIST.GET_NODE(SMALLEST);
         FIND_SMALLER: loop
           exit FIND_SMALLER when RESULT_LIST.GET_NODE(RESULTS_POINTER) < DIS-
TANCE;
      RESULTS_POINTER := RESULT_LIST.NEXT_LINK(RESULTS_POINTER);
      NEXT_WORD(WORD,WORD_POINTER);
         end loop FIND_SMALLER;
    TEMP_LIST.REMOVE_NODE(TEMP_LIST.GET_NODE(SMALL-
EST),TEMP_HEAD,TEMP_TAIL);
    TEMP_LIST.ADD_TO(RESULT_LIST.GET_NODE(RE-
SULTS_POINTER),TEMP_HEAD,TEMP_TAIL);
    RESULTS_POINTER := RESULT_LIST.NEXT_LINK(RESULTS_POINTER);
    DATABASE_LIST.REMOVE_NODE(DATABASE_LIST.GET_NODE(SMALLEST_IN_SUB -
LIST),SUBLIST_HEAD,SUBLIST_TAIL);
    DATABASE_LIST.ADD_TO(DATABASE_LIST.GET_NODE(WORD_POINTER),SUB -
LIST_HEAD,SUBLIST_TAIL);
    NEXT_WORD(WORD,WORD_POINTER);
       end loop;
     exception
       when RESULT_LIST.UNDER_FLOW =>
         NULL;
     end;
   end CREATE_SUBLIST;

   function AGREE return FLOAT is
   - [function_header_comment]
     RESULTS : RESULT_RECORD_TYPE;
     FIRST_ANSWER, SMALLEST, DISAGREE, COUNTER : INTEGER := 0;

     FUSION_POINTER : FUSION_LIST.LIST := FUSION_HEAD;
   begin
     RESULTS := GET_NODE(FUSION_POINTER);
     FIRST_ANSWER := SMALL_VALUE_BY_INDEX(RESULTS);
     COUNTER := 1;
    loop
      FUSION_POINTER := NEXT_LINK(FUSION_POINTER);
      RESULTS := GET_NODE(FUSION_POINTER);
      COUNTER := COUNTER + 1;
      SMALLEST := SMALL_VALUE_BY_INDEX(RESULTS);
       if SMALLEST /= FIRST_ANSWER then
         DISAGREE := DISAGREE + 1;
       end if;
    end loop;
  exception
    when FUSION_LIST.UNDER_FLOW =>
      return FLOAT(COUNTER-DISAGREE)/FLOAT(COUNTER);
  end AGREE;

  procedure FUSION_CONTROL_PANEL is
  - [procedure_header_comment]
    type CHOICE_INTEGER is range 1..11;
    CHOICE : CHOICE_INTEGER;
    NUM_WORDS : INTEGER;

    package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (OUTPUT_CHAN-
NEL_TYPE);
    package FLOAT_IO is new TEXT_IO.FLOAT_IO (FLOAT);
    package INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
    package REG_INTEGER_IO is new TEXT_IO.INTEGER_IO (INTEGER);
    use INTEGER_IO, REG_INTEGER_IO, ENUMERATION_IO, FLOAT_IO;

  begin
    CREATE (FILE => REPORT_FILE);
    MAIN: loop
      begin
    NEW_LINE(2);
    PUT_LINE (ITEM =>     1 - SET OUTPUT CHANNEL (SCREEN OR FILE)");
    PUT_LINE (ITEM => "   2 - SET FILE NAME FOR RESULTS");
    PUT_LINE (ITEM => "   3 - CREATE SUBLIST");
    PUT_LINE (ITEM => "   4 - OUTPUT RESULTS FOR A FEATURE");
    PUT_LINE (ITEM => "   5 - OUTPUT RESULTS FOR ALL FEATURES");
    PUT_LINE (ITEM => "   6 - OUTPUT SMALLEST DTW VALUE FOR A FEATURE");
    PUT_LINE (ITEM => "   7 - OUTPUT SMALLEST DTW VALUE FOR ALL FEATURES");
    PUT_LINE (ITEM => "   8 - PERFORM FUSION");
    PUT_LINE (ITEM => "   9 - FUSION RESULT");
    PUT_LINE (ITEM => "  10 - CLEAR FUSION LIST");
PUT_LINE (ITEM => "   11 - QUIT");
NEW_LINE;
PUT (ITEM => "    CHOICE -> ");
GET (ITEM => CHOICE);
GET_LINE (ITEM => DUMMY);
case CHOICE is
  when 1 =>
NEW_LINE;
PUT_LINE (ITEM => "TYPE OUTPUT CHANNEL NAME, SCREEN OR FILE");
GET (ITEM => OUTPUT_CHANNEL);
GET_LINE (ITEM => DUMMY);
  when 2 =>
CLOSE (FILE => REPORT_FILE);
NEW_LINE;
PUT_LINE (ITEM => "TYPE NAME OF FILE");
GET_LINE (ITEM => THE_FILE);
CREATE (
   FILE => REPORT_FILE,
   MODE => OUT_FILE,
   NAME => THE_FILE);
  when 3 =>
         PUT_LINE (ITEM => "TYPE THE NUMBER OF WORDS TO BE IN THE SUBLIST");
         REG_INTEGER_IO.GET (ITEM => NUM_WORDS);
GET_LINE (ITEM => DUMMY);
CREATE_SUBLIST(NUM_WORDS);
  when 4 =>
FEATURE_RESULTS_OUTPUT(OUTPUT_CHANNEL,REPORT_FILE);
  when 5 =>
ALL_FEATURE_RESULTS_OUTPUT(OUTPUT_CHANNEL,REPORT_FILE);
  when 6 =>
FEATURE_SMALLEST_RESULT(OUTPUT_CHANNEL,REPORT_FILE);
  when 7 =>
ALL_FEATURE_SMALLEST_RESULT(OUTPUT_CHANNEL,REPORT_FILE);
  when 8 =>
FUSION;
  when 9 =>
FUSION_RESULT;
  when 10 =>
CLEAR_LIST(FUSION_HEAD, FUSION_TAIL);
  when 11 =>
exit MAIN;
end case;
```

```
        exception
      when DATA_ERROR =>
        PUT_LINE (ITEM => "THERE ARE 9 CHOICES, SO TYPE A NUMBER BETWEEN 1 AND 9");
        end;
      end loop MAIN;
      CLOSE (FILE => REPORT_FILE);
    end FUSION_CONTROL_PANEL;

  end FUSION_PACKAGE;
```

```
- [source_file_header_comment]
with LINKED_LIST, STRING_OPERATIONS, DATABASE_PACKAGE;
use STRING_OPERATIONS, DATABASE_PACKAGE;
package FUSION_REC_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    THIS PACKAGE CONTAINS TYPE AND LIST DEFINITIONS TO MAKE THE FUSION
-  PACKAGE AND THE RECOGNITION PACKAGE WORK TOGETHER
-
- [optional package tags]
- -

  - LIST FOR STORING DISTANCE VALUES FROM THE RECOGNIZERS
  package RESULT_LIST is new LINKED_LIST (ITEM_TYPE => FLOAT);
  use RESULT_LIST;

  - TELLS THE FUSION PACKAGE WHERE TO OUTPUT THE RESULTS FROM ITS REPORTS
  type OUTPUT_CHANNEL_TYPE is (SCREEN, FILE);
  - A RECORD FOR STORING THE RESULTS FROM A RECOGNIZER ON A FEATURE
  type RESULT_RECORD_TYPE is
    record
      RESULT_HEAD : RESULT_LIST.LIST;
      RESULT_TAIL : RESULT_LIST.LIST;
      FEATURE_NAME : VAR_STRING;
      FUSION_FACTOR : FLOAT := 1.0;
    end record;
  - RECORD FOR CREATING FUSED FEATURES
  type FEATURE_RECORD_TYPE is
    record
      NAME : VAR_STRING;
      COMPONENTS : INTEGER;
    end record;

  - LIST FOR STORING THE RESULTS FOR RECOGNIZERS ON DIFFERENT FEATURE
  package FUSION_LIST is new LINKED_LIST (ITEM_TYPE => RESULT_RECORD_TYPE);
  - LIST FOR CREATING FUSED FEATURES
  package FUSED_FEATURE_LIST is new LINKED_LIST (ITEM_TYPE => FEATURE_RE-
CORD_TYPE);
  - LIST FOR STORING USER PREFIXES TO BE TEMPLATED
  package PREFIX_LIST is new LINKED_LIST (ITEM_TYPE => VAR_STRING);
  use FUSED_FEATURE_LIST, FUSION_LIST, PREFIX_LIST;

  FUSION_HEAD, FUSION_TAIL : FUSION_LIST.LIST;
  PREFIX_HEAD, PREFIX_TAIL : PREFIX_LIST.LIST;

end FUSION_REC_PACKAGE;
```

```
- [source_file_header_comment]
with TEXT_IO, STRING_OPERATIONS, LINKED_LIST;
use TEXT_IO, STRING_OPERATIONS;
package DATABASE_PACKAGE is
- ++
-
- PACKAGE DESCRIPTION:
-
-    The database_package contains the packages and functions for
- manipulating the word name list. This list holds all the names of the
- words in the database
-
- [optional package tags]
- -

  - instantiate linked list for the database list
  package DATABASE_LIST is new LINKED_LIST (ITEM_TYPE => VAR_STRING);
  - instantiate linked_list foe the speech list
  package SPEECH_LIST is new LINKED_LIST (ITEM_TYPE => FLOAT);
  use SPEECH_LIST, DATABASE_LIST;

  type LIST_TYPE is (LIST, SUBLIST);
  LIST_POINTER : LIST_TYPE := LIST;
  LIST_HEAD, LIST_TAIL, SUBLIST_HEAD, SUBLIST_TAIL, DATABASE_HEAD, DATA-
BASE_TAIL : DATABASE_LIST.LIST;
  SPEECH_HEAD, SPEECH_TAIL : SPEECH_LIST.LIST;
  COUNT_STRING, DIRECTORY_NAME, LIBRARY_PREFIX : VAR_STRING;
  NUMBER_OF_WORDS : INTEGER := 0;
  WORDS_FILE : FILE_TYPE;
  WORDS_FILE_NAME : VAR_STRING;

  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    This procedure sets the path ot the database
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure SET_DIRECTORY_NAME (NEW_DIR : in VAR_STRING);
```

```
- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure sets the library prefix of the words in the database
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure SET_LIBRARY_PREFIX (NEW_PRE : in VAR_STRING);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure is used to reset the word count in case it becomes
-  incorrect
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure SET_WORD_COUNT (COUNT : in INTEGER);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure adds a word name to the database list
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure ADD_LIBRARY_WORD (NEW_WORD : in VAR_STRING);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
```

```
-    This procedure deletes a word name from the database list
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure DELETE_LIBRARY_WORD (THE_WORD : in VAR_STRING);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure reset the database list after words are added
-  or deleted
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure RESET_DATABASE_LIST;

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure returns the name of the next word in the database
-  list and increments the pointer
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure NEXT_WORD (THE_WORD : out VAR_STRING; POINTER : in out DATA-
BASE_LIST.LIST);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This produres provides interactive control of the database
-  functions directly from the menu
-
```

```
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure DATABASE_CONTROL_PANEL (UPDATE : in out BOOLEAN );

  - ++
  -
  - FUNCTIONAL DESCRIPTION:
  -
  -    THis procedure is used to toggle between using the database list
  -  as the entire list or as a sublist
  -
  - [formal parameters]
  - [design]
  - [optional subprogram tags]
  - -
  procedure SET_LIST_POINTER (TO_LIST : in LIST_TYPE);

  package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (LIST_TYPE);
  USE ENUMERATION_IO;

end DATABASE_PACKAGE;
```

```
package body DATABASE_PACKAGE is
- [package_body_header_comment]

  procedure SET_DIRECTORY_NAME (NEW_DIR : in VAR_STRING) is
  - [procedure_header_comment]
  begin
    DIRECTORY_NAME := NEW_DIR;
  end SET_DIRECTORY_NAME;

  procedure SET_LIBRARY_PREFIX (NEW_PRE : in VAR_STRING) is
  - [procedure_header_comment]
  begin
    LIBRARY_PREFIX := NEW_PRE;
  end SET_LIBRARY_PREFIX;

  procedure SET_WORD_COUNT (COUNT : in INTEGER) is
  - [procedure_header_comment]
  begin
    NUMBER_OF_WORDS := COUNT;
  end SET_WORD_COUNT;

  procedure ADD_LIBRARY_WORD (NEW_WORD : in VAR_STRING) is
  - [procedure_header_comment]
  begin
    if LIST_POINTER = LIST then
      DATABASE_LIST.ADD_TO(NEW_WORD,LIST_HEAD,LIST_TAIL);
    else
      DATABASE_LIST.ADD_TO(NEW_WORD,SUBLIST_HEAD,SUBLIST_TAIL);
    end if;
    NUMBER_OF_WORDS := NUMBER_OF_WORDS + 1;
  end ADD_LIBRARY_WORD;

  procedure DELETE_LIBRARY_WORD (THE_WORD : in VAR_STRING) is
  - [procedure_header_comment]
  begin
    if LIST_POINTER = LIST then
      DATABASE_LIST.REMOVE_NODE(THE_WORD,LIST_HEAD,LIST_TAIL);
    else
      DATABASE_LIST.REMOVE_NODE(THE_WORD,LIST_HEAD,LIST_TAIL);
    end if;
    NUMBER_OF_WORDS := NUMBER_OF_WORDS - 1;
  end DELETE_LIBRARY_WORD;
  procedure NEXT_WORD (THE_WORD : out VAR_STRING; POINTER : in out DATA-
BASE_LIST.LIST) is
  - [procedure_header_comment]
  begin
    THE_WORD := GET_NODE(POINTER);
   POINTER := NEXT_LINK(POINTER);
  end NEXT_WORD;

  procedure RESET_DATABASE_LIST is
  - [procedure_header_comment]
    WORDS : VAR_STRING;
  begin
   DATABASE_LIST.CLEAR_LIST(LIST_HEAD,LIST_TAIL);
   ASSIGN(WORDS_FILE_NAME,"DATABASE.DAT");
   WORDS_FILE_NAME := DIRECTORY_NAME&WORDS_FILE_NAME;
   OPEN (
     FILE => WORDS_FILE,
     MODE => IN_FILE,
     NAME => WORDS_FILE_NAME);

    begin
     loop
   STRING_OPERATIONS.GET_LINE (FILE => WORDS_FILE , ITEM => WORDS);
   ADD_TO(WORDS,LIST_HEAD,LIST_TAIL);
   NUMBER_OF_WORDS := NUMBER_OF_WORDS + 1;
     end loop;
    exception
      when END_ERROR =>
        CLOSE (FILE => WORDS_FILE);
    end;
  end RESET_DATABASE_LIST;

  procedure SET_LIST_POINTER (TO_LIST : in LIST_TYPE) is
  - [procedure_header_comment]
  begin
   LIST_POINTER := TO_LIST;
    if TO_LIST = LIST then
      DATABASE_HEAD := LIST_HEAD;
     DATABASE_TAIL := LIST_TAIL;
    else
      DATABASE_HEAD := SUBLIST_HEAD;
    DATABASE_TAIL := SUBLIST_TAIL;
  end if;
end SET_LIST_POINTER;

procedure DATABASE_CONTROL_PANEL (UPDATE : in out BOOLEAN) is
- [procedure_header_comment]

  type CHOICE_INTEGER is range 1..7;
  DUMMY, THE_WORD : VAR_STRING;
  CHOICE : CHOICE_INTEGER;
  ANSWER : CHARACTER;
  WORD_POINTER : DATABASE_LIST.LIST;

   package ENUMERATION_IO is new TEXT_IO.ENUMERATION_IO (LIST_TYPE);
  package INTEGER_IO is new TEXT_IO.INTEGER_IO (CHOICE_INTEGER);
  use INTEGER_IO, ENUMERATION_IO;
begin
  NULL_POINTER(WORD_POINTER);
  UPDATE := FALSE;
  MAIN: loop
    begin
  NEW_LINE (2);
  PUT_LINE (ITEM => "     1 - ADD WORD(S) TO DATABASE");
  PUT_LINE (ITEM => "     2 - DELETE WORDS FROM DATABASE");
  PUT_LINE (ITEM => "     3 - LIST LIBRARY WORDS");
  PUT_LINE (ITEM => "     4 - SET DIRECTORY NAME");
  PUT_LINE (ITEM => "     5 - SET LIBRARY PREFIX");
  PUT_LINE (ITEM => "     6 - SET LIST POINTER");
  PUT_LINE (ITEM => "     7 - QUIT EXTRACTION OPERATIONS");
  NEW_LINE;
  PUT ("     CHOICE -> ");
  GET (ITEM => CHOICE);
  GET_LINE (ITEM => DUMMY);
  case CHOICE is
    when 1 =>
  ADD_WORD: loop
    PUT_LINE (ITEM => "TYPE IN WORD TO ADD TO THE DATABASE");
    GET_LINE (ITEM => THE_WORD);
    ADD_LIBRARY_WORD(THE_WORD);
    PUT_LINE (ITEM => "DO YOU WNAT TO ADD ANOTHER?");
    GET (ITEM => ANSWER);
    GET_LINE (ITEM => THE_WORD);
  exit ADD_WORD when NOT (ANSWER = 'Y' OR ANSWER = 'y');
end loop ADD_WORD;
UPDATE := TRUE;
  when 2 =>
DELETE_WORD: loop
  PUT_LINE (ITEM => "TYPE IN WORD TO DELETE FROM  THE DATABASE");
  GET_LINE (ITEM => THE_WORD);
  DELETE_LIBRARY_WORD(THE_WORD);
  PUT_LINE (ITEM => "DO YOU WNAT TO DELETE ANOTHER?");
  GET (ITEM => ANSWER);
  GET_LINE (ITEM => THE_WORD);
  exit DELETE_WORD when NOT (ANSWER = 'Y' OR ANSWER = 'y');
end loop DELETE_WORD;
UPDATE := TRUE;
  when 3 =>
begin
  WORD_POINTER := DATABASE_HEAD;
  NEXT_WORD(THE_WORD,WORD_POINTER);
  loop
PUT_LINE (ITEM => THE_WORD);
NEXT_WORD(THE_WORD,WORD_POINTER);
  end loop;
exception
  when DATABASE_LIST.UNDER_FLOW =>
NULL;
end;
  when 4 =>
NEW_LINE;
PUT_LINE (ITEM => "TYPE IN NEW DIRECTORY NAME");
GET_LINE (ITEM => THE_WORD);
SET_DIRECTORY_NAME(THE_WORD);
UPDATE := TRUE;
  when 5 =>
NEW_LINE;
PUT_LINE (ITEM => "TYPE IN NEW LIBRARY PREFIX");
GET_LINE (ITEM => THE_WORD);
SET_LIBRARY_PREFIX(THE_WORD);
UPDATE := TRUE;
  when 6 =>
          PUT_LINE (ITEM => "TYPE LIST OR SUBLIST");
          ENUMERATION_IO.GET (ITEM => LIST_POINTER);
GET_LINE (ITEM => THE_WORD);
```

```
      when 7 =>
             if UPDATE = TRUE then
      WORDS_FILE_NAME := DIRECTORY_NAME&"DATABASE.DAT";
      OPEN (
    FILE => WORDS_FILE,
    MODE => OUT_FILE,
    NAME => WORDS_FILE_NAME);
      WORD_POINTER := DATABASE_HEAD;
      begin
    loop
      NEXT_WORD(THE_WORD,WORD_POINTER);
      STRING_OPERATIONS.PUT_LINE (FILE => WORDS_FILE , ITEM => THE_WORD);
    end loop;
      exception
    when DATABASE_LIST.UNDER_FLOW =>
      CLOSE (FILE => WORDS_FILE);
      end;
             end if;
    exit MAIN;
    end case;
      exception
    when DATA_ERROR =>
      PUT_LINE (ITEM => "OOOCH! HEY, WATCH WHAT YOUR TYPING - I'LL PULL A CON-
STRAINT");
      end;
    end loop MAIN;
  end DATABASE_CONTROL_PANEL;

begin
  ASSIGN(DIRECTORY_NAME,"[TRATHBUN.SPEECH.DATABASE]");
  ASSIGN(LIBRARY_PREFIX,"");
  RESET_DATABASE_LIST;
  SET_LIST_POINTER(LIST);
  SPEECH_LIST.NULL_POINTER(SPEECH_HEAD);
  SPEECH_LIST.NULL_POINTER(SPEECH_TAIL);
end DATABASE_PACKAGE;
```

```
- [source_file_header_comment]
with UNCHECKED_DEALLOCATION;
generic
   type ITEM_TYPE is private;
package LINKED_LIST is
- ++
-
- PACKAGE DESCRIPTION:
-
-    THIS PACKAGE CONTAINS THE PROCEDURES AND FUNCTIONS FOR IMPLEMENTING
-  A GENERIC LINKED LIST.  THIS LINKED LIST ACTS LIKE A QUEUE IN THE NEW
-  NODES ARE ALWAYS ADDED TO THE TAIL AND USUALLY TAKEN OFF THE FRONT.
-  HOWEVER NODES MAY BE ACCESS OR DELETED FROM THE MIDDLE.  THIS PACKAGE
-  FREES UP MEMORY TAKEN BY DELETED NODE.
-
- [optional package tags]
- -

   type LIST is private;

   - ++
   -
   - FUNCTIONAL DESCRIPTION:
   -
   -    THIS PROCEDURE ADDS A NODE TO THE TAIL OF THE LIST
   -
   - [formal parameters]
   - [design]
   - [optional subprogram tags]
   - -
   procedure ADD_TO (THE_ITEM : in ITEM_TYPE;
         THE_LIST, THE_TAIL : in out LIST);

   procedure ADD_TO_REVERSE (THE_ITEM : in ITEM_TYPE;
              THE_LIST, THE_TAIL : in out LIST);

   - ++
   -
   - FUNCTIONAL DESCRIPTION:
   -
   -    THE FUNCTION RETURNS THE VALUE OF THE NODE POINTED TO BY THE
   -  INPUT POINTER
```

```
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
function GET_NODE (THE_POINTER : in LIST) return ITEM_TYPE;

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE DELETE THE NODE AT THE HEAD OF THE LIST AND RETURN
-  THE VALUE OF THAT NODE
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure REMOVE_NEXT_NODE (THE_ITEM : out ITEM_TYPE;
           THE_LIST, THE_TAIL: in out LIST);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE DELETE THE NODE IN THE LIST WHOS VALUE IS THE SAME
-  AS THE INPUT VALUE
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure REMOVE_NODE (THE_ITEM : in ITEM_TYPE;
     THE_LIST, THE_TAIL: in out LIST);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE SETS THE LIST POINTERS TO NULL AND FREES THE MEMORY
-  TAKEN UP BY ALLNODE PREVIOUSLY IN THE LIST
-
```

```
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure CLEAR_LIST (LIST_HEAD, LIST_TAIL : in out LIST);

- [procedure_header_comment]
procedure NULL_POINTER (LIST_POINTER : out LIST);

- ++
- FUNCTIONAL DESCRIPTION:
-
-    THIS FUNCTION INCREMENTS A GIVEN POINTER
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    {tbs}
-
- -
function NEXT_LINK (THE_POINTER : in LIST) return LIST;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    THIS FUNCTION DETERMINES IF THE GIVEN POINTER IS EQUAL TO THE
-  VALUE NULL.  IF TRUE A BOOLEAN TRUE IS RETURN, FALSE OTHERWISE
-
- [formal parameters]
- {design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    {tbs}
-
- -
- [procedure_header_comment]
procedure REVERSE_LIST (HEAD, TAIL : in out LIST);
function IS_NULL (THE_LIST_POINTER : in LIST) return BOOLEAN;
```

```
- ++
- FUNCTIONAL DESCRIPTION:
-
-    THIS FUNCTIONS DETERMINES IF TWO POINTERS OF TYPE LIST ARE EQUAL
-  AND RETURNS THE APPROIATE BOOLEAN ANSWER
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    {tbs}
-
- -
function IS_EQUAL (POINTER_ONE, POINTER_TWO : in LIST) return BOOLEAN;

generic
   with procedure PROCESS (ITEM_ONE, ITEM_TWO : in out ITEM_TYPE;
                   CONTINUE : out BOOLEAN);
- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    THIS PROCEDURE IS A GENERIC PROCEDURE USE FOR TRAVERSING A LIST
-  TO LOOK FOR A SPECIFIC ELEMENT OF A NODE. THE USER INSTANTIATES THIS
-  PROCEDURE WITH A PROCEDURE THAT DETERMINES WHEN A MATCH OCCURS
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure ITERATE_LIST (THE_ITEM : in out ITEM_TYPE;
     THE_LIST, THE_TAIL : in LIST);

- [procedure_header_comment]
procedure COPY_LIST (HEAD1, TAIL1, HEAD2, TAIL2 : in out LIST);

OVER_FLOW : exception;
NOT_IN_LIST : exception;
UNDER_FLOW : exception;

private
```

```
  type NODE;
  type LIST is access NODE;
  type NODE is
    record
      THE_ITEM : ITEM_TYPE;
      NEXT : LIST;
    end record;

  - THIS INSTANTATED PROCEDURE FREES UP USED MEMORY AFTER A NODE IS DELETED
  procedure FREE is new UNCHECKED_DEALLOCATION (OBJECT => NODE,
                                                NAME => LIST);

end LINKED_LIST;
```

```
- [source_file_header_comment]
package body LINKED_LIST is
- [package_body_header_comment]
-  [basic_declarative_item]...
  procedure ADD_TO (THE_ITEM : in ITEM_TYPE; THE_LIST, THE_TAIL : in out LIST) is
  - [procedure_header_comment]
    TEMP_POINTER : LIST;
  begin
    TEMP_POINTER := new NODE'(THE_ITEM => THE_ITEM,
     NEXT     => NULL);
    if THE_LIST = NULL then
      THE_LIST := TEMP_POINTER;
      THE_TAIL := THE_LIST;
    else
      THE_TAIL.NEXT := TEMP_POINTER;
      THE_TAIL := TEMP_POINTER;
    end if;
  exception
    when STORAGE_ERROR =>
      raise OVER_FLOW;
  end ADD_TO;
  function GET_NODE (THE_POINTER : in LIST) return ITEM_TYPE is
  - [function_header_comment]
  begin
    if THE_POINTER = NULL then
      raise UNDER_FLOW;
    else
      return THE_POINTER.THE_ITEM;
    end if;
  end GET_NODE;
  procedure REMOVE_NEXT_NODE (THE_ITEM : out ITEM_TYPE;
                  THE_LIST, THE_TAIL : in out LIST) is
  - [procedure_header_comment]
    DUMMY_LIST : LIST;
  begin
    if THE_LIST = NULL then
      raise UNDER_FLOW;
    elsif THE_LIST = THE_TAIL then
      THE_ITEM := THE_LIST.THE_ITEM;
      FREE(THE_LIST);
      THE_TAIL := NULL;
    else
```

```
      THE_ITEM := THE_LIST.THE_ITEM;
      DUMMY_LIST := THE_LIST;
      THE_LIST := THE_LIST.NEXT;
      FREE(DUMMY_LIST);
    end if;
end REMOVE_NEXT_NODE;

procedure REMOVE_NODE (THE_ITEM : in ITEM_TYPE;
      THE_LIST, THE_TAIL : in out LIST) is
- [procedure_header_comment]
    DUMMY_LIST, POINTER : LIST := THE_LIST;
    TRAIL_POINTER : LIST := NULL;
begin
    if THE_LIST = NULL then
      raise UNDER_FLOW;
    elsif THE_LIST = THE_TAIL then
      FREE(THE_LIST);
      THE_TAIL := NULL;
    else
      ITERATE: loop
        exit ITERATE when POINTER.THE_ITEM = THE_ITEM;
    TRAIL_POINTER := POINTER;
    POINTER := POINTER.NEXT;
        if POINTER = NULL then
          raise NOT_IN_LIST;
        end if;
      end loop ITERATE;
      if TRAIL_POINTER = NULL then
        THE_LIST := THE_LIST.NEXT;
      elsif POINTER = THE_TAIL then
        THE_TAIL := TRAIL_POINTER;
    FREE(POINTER);
      else
    DUMMY_LIST := TRAIL_POINTER.NEXT;
        TRAIL_POINTER.NEXT := POINTER.NEXT;
    FREE(DUMMY_LIST);
      end if;
    end if;
end REMOVE_NODE;

procedure CLEAR_LIST (LIST_HEAD, LIST_TAIL : in out LIST) is
- [procedure_header_comment]
```

```
    DEALLOCATE_POINTER : LIST := LIST_HEAD;
    DUMMY_POINTER : LIST;
  begin
-   while DEALLOCATE_POINTER /= NULL loop
.     DUMMY_POINTER := DEALLOCATE_POINTER;
-     DEALLOCATE_POINTER := DEALLOCATE_POINTER.NEXT;
-     FREE(DUMMY_POINTER);
-   end loop;
    LIST_HEAD := NULL;
    LIST_TAIL := NULL;
  end CLEAR_LIST;

  procedure NULL_POINTER (LIST_POINTER : out LIST) is
  - [procedure_header_comment]
  begin
    LIST_POINTER := NULL;
  end NULL_POINTER;

  function NEXT_LINK (THE_POINTER : in LIST) return LIST is
  - [function_header_comment]
  begin
    if THE_POINTER = NULL then
      raise UNDER_FLOW;
    else
      return THE_POINTER.NEXT;
    end if;
  end NEXT_LINK;

  procedure ADD_TO_REVERSE (THE_ITEM : in ITEM_TYPE; THE_LIST, THE_TAIL : in out LIST)
is
  - [procedure_header_comment]
    TEMP_POINTER : LIST;
  begin
    TEMP_POINTER := new NODE'(THE_ITEM => THE_ITEM,
     NEXT     => NULL);
    if THE_LIST = NULL then
      THE_LIST := TEMP_POINTER;
      THE_TAIL := THE_LIST;
    else
      TEMP_POINTER.NEXT := THE_LIST;
      THE_LIST := TEMP_POINTER;
    end if;
```

```
exception
  when STORAGE_ERROR =>
    raise OVER_FLOW;
end ADD_TO_REVERSE;

procedure REVERSE_LIST (HEAD, TAIL : in out LIST) is
- [procedure_header_comment]
  REV_HEAD, REV_TAIL : LIST := NULL;
  THIS_ITEM : ITEM_TYPE;
begin
  loop
    REMOVE_NEXT_NODE(THIS_ITEM,HEAD,TAIL);
    ADD_TO_REVERSE(THIS_ITEM,REV_HEAD,REV_TAIL);
  end loop;
exception
  when UNDER_FLOW =>
    HEAD := REV_HEAD;
    TAIL := REV_TAIL;
end REVERSE_LIST;

function IS_NULL (THE_LIST_POINTER : in LIST) return BOOLEAN is
- [function_header_comment]
  FLAG : BOOLEAN := FALSE;
begin
  if THE_LIST_POINTER = NULL then
    FLAG := TRUE;
  end if;
  return FLAG;
end IS_NULL;

function IS_EQUAL (POINTER_ONE, POINTER_TWO : in LIST) return BOOLEAN is
- [function_header_comment]
begin
  if POINTER_ONE = POINTER_TWO then
    return TRUE;
  else
    return FALSE;
  end if;
end IS_EQUAL;

procedure COPY_LIST (HEAD1, TAIL1, HEAD2, TAIL2 : in out LIST) is
- [procedure_header_comment]
```

```
    LIST_POINTER : LIST := HEAD1;
    THE_ITEM : ITEM_TYPE;
  begin
    loop
      THE_ITEM := GET_NODE(LIST_POINTER);
      LIST_POINTER := NEXT_LINK(LIST_POINTER);
      ADD_TO(THE_ITEM, HEAD2, TAIL2);
    end loop;
  exception
    when UNDER_FLOW =>
      NULL;
  end COPY_LIST;

  procedure ITERATE_LIST (THE_ITEM : in out ITEM_TYPE;
             THE_LIST, THE_TAIL : in LIST) is
  - [procedure_header_comment]
    POINTER : LIST := THE_LIST;
    CONTINUE : BOOLEAN := TRUE;
  begin
    while CONTINUE = TRUE loop
      if THE_LIST = NULL OR POINTER = NULL then
        raise NOT_IN_LIST;
      end if;
      PROCESS(POINTER.THE_ITEM,THE_ITEM,CONTINUE);
      POINTER := POINTER.NEXT;
    end loop;
  end ITERATE_LIST;
-  [later_declarative_item]...
-[executable_part]
end linked_LIST;
```

```
WITH TEXT_IO;

PACKAGE STRING_OPERATIONS IS
-------------------------------------
- THE PURPOSE OF THIS PACKAGE IS TO DEFINE THE VARIABLE STRING DATA TYPE AND
- PROVIDE THE VARIABLE STRING INPUT, OUTPUT, AND EQUALITY FUNCTIONS.
-
-
- DATE: 5 SEP 90
- VERSION: 3
- AUTHOR: THOMAS RATHBUN
-
- TITLE: VARIABLE STRING DATA TYPE AND I/O AND EQUALITY OPERATIONS
- FILENAME: STRING_OPERATIONS.ADA
- OPERATING SYSTEM: VMS 5.3
- LANGUAGE: ADA
-
- THIS FILE IS A SELF STANDING COMPILABLE UNIT.
-
- THE PACKAGE CONTAINS 2 PROCEDURES FOR INPUT AND OUTPUTING DATA OF
- VAR_STRING TYPE AND 1 FUNCTION FOR DETERMINING EQUALITY
--------------------------------------

   type VAR_STRING is private;

   - PROCEDURE PUT IS FOR OUTPUTING DATA OF TYPE VAR_STRING
   PROCEDURE PUT (ITEM: IN VAR_STRING);

   - PROCEDURE PUT IS FOR OUTPUTING DATA OF TYPE VAR_STRING
   PROCEDURE PUT (FILE : in out TEXT_IO.FILE_TYPE; ITEM: IN VAR_STRING);

   - PROCEDURE PUT IS FOR OUTPUTING DATA OF TYPE VAR_STRING
   PROCEDURE PUT_LINE (ITEM: IN VAR_STRING);

   - [procedure_header_comment]
   procedure PUT_LINE (FILE : in out TEXT_IO.FILE_TYPE;
             ITEM : in VAR_STRING);

   - PROCEDURE GET_LINE IS FOR INPUTING DATA OF TYPE VAR_STRING
   PROCEDURE GET_LINE(ITEM: OUT VAR_STRING);

   - ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procdure reads a text line from a file and stores it
- in a var_string variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
--
procedure GET_LINE (FILE : in out TEXT_IO.FILE_TYPE;
              ITEM : out VAR_STRING);

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure opens a file whos filename is a var_string type
-
- [formal parameters]
- [design]
- [optional subprogram tags]
--
procedure OPEN (FILE : in out TEXT_IO.FILE_TYPE;
           MODE : in TEXT_IO.FILE_MODE;
           NAME : in VAR_STRING;
           FORM : in STRING := "");

- ++
-
- FUNCTIONAL DESCRIPTION:
-
-    This procedure creates a file who's filename is a var_string
- type
-
- [formal parameters]
- [design]
- [optional subprogram tags]
--
procedure CREATE (FILE : in out TEXT_IO.FILE_TYPE;
 MODE : in TEXT_IO.FILE_MODE;
            NAME : in VAR_STRING;
            FORM : in STRING := "");


- ++
-
- FUNCTIONAL DESCRIPTION:
-
-     This procedure stores a text string in a var_string variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- -
procedure ASSIGN (LEFT_VAR_STRING : out VAR_STRING; RIGHT_STRING : in STRING);

- ++
- FUNCTIONAL DESCRIPTION:
-
-     This function compares to var_string variables
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-     true or false
-
- -
FUNCTION IS_EQUAL (LEFT, RIGHT : in VAR_STRING) return BOOLEAN;

- ++
- FUNCTIONAL DESCRIPTION:
-
-     This function compares a var_string variable with s string
-  variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-     true or false
```

```
-
- -
FUNCTION IS_EQUAL (LEFT : in VAR_STRING; RIGHT : in STRING) return BOOLEAN;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    This function concatenates two var_string variables
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    var_string
-
- -
function "&" (LEFT, RIGHT : in VAR_STRING) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    This function concatenates a string variable and a var_string
- variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    var_string
-
- -
function "&" (LEFT : in STRING; RIGHT : in VAR_STRING) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    This function concatenates a var_string variable and a string
- varable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-     var_string
-
- -
function "&" (LEFT : in VAR_STRING; RIGHT : in STRING) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-     This function concatenates two string variables
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-     var_string
-
- -
function "&" (LEFT, RIGHT : in STRING) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-     This function concatenates a var_string variable and a
-  character vaiable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-     var_string
-
- -
function "&" (LEFT : in VAR_STRING; RIGHT : in CHARACTER) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
```

```
-
-    This funciton concatenates a string variable and a character
- variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    var_string
-
- -
function "&" (LEFT : in STRING; RIGHT : in CHARACTER) return VAR_STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    This function returns the string value of a var_string variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    string
-
- -
function STRING_VALUE (A_STRING : in VAR_STRING) return STRING;

- ++
- FUNCTIONAL DESCRIPTION:
-
-    This function returns the character at a position in a
- var_string variable
-
- [formal parameters]
- [design]
- [optional subprogram tags]
- RETURN VALUE:
-
-    character
-
```

```
    - -
    function STRING_VALUE (POSITION : INTEGER; A_STRING : in VAR_STRING) return CHAR-
ACTER;

    - ++
    - FUNCTIONAL DESCRIPTION:
    -
    -    This function returns the string length of a var_string variable
    -
    - [formal parameters]
    - [design]
    - [optional subprogram tags]
    - RETURN VALUE:
    -
    -    {tbs}
    -
    - -
    function STRING_LENGTH (A_STRING : in VAR_STRING) return INTEGER;

private
  - DECLARE DATA TYPE VAR_STRING
  TYPE VAR_STRING IS
      RECORD
    THE_LENGTH: NATURAL := 0; - number of characters in string
    THE_ITEMS: STANDARD.STRING (1..80) := (' ', OTHERS => ' ');
      END RECORD;

BLANK_STRING : VAR_STRING;
END STRING_OPERATIONS;
```

```
PACKAGE BODY STRING_OPERATIONS IS

    PROCEDURE PUT (ITEM: IN VAR_STRING) IS
-------------------------------------
- THE PURPOSE OF THIS PROCEDURE IS TO PRINT THE DATA CONTAINED IN A VAR_STRING
- DATA TYPE
-------------------------------------
    BEGIN

    - OUTPUT DATA IN A VAR_STRING DATA TYPE VARIABLE
    TEXT_IO.PUT(ITEM.THE_ITEMS(1..ITEM.THE_LENGTH));

    END PUT;

    PROCEDURE PUT (FILE : in out TEXT_IO.FILE_TYPE; ITEM: IN VAR_STRING) IS
-------------------------------------
- THE PURPOSE OF THIS PROCEDURE IS TO PRINT THE DATA CONTAINED IN A VAR_STRING
- DATA TYPE
-------------------------------------
    BEGIN

    - OUTPUT DATA IN A VAR_STRING DATA TYPE VARIABLE
    TEXT_IO.PUT(FILE => FILE, ITEM => ITEM.THE_ITEMS(1..ITEM.THE_LENGTH));

    END PUT;

    PROCEDURE PUT_LINE (ITEM: IN VAR_STRING) IS
-------------------------------------
- THE PURPOSE OF THIS PROCEDURE IS TO PRINT THE DATA CONTAINED IN A VAR_STRING
- DATA TYPE
-------------------------------------
    BEGIN

    - OUTPUT DATA IN A VAR_STRING DATA TYPE VARIABLE
    TEXT_IO.PUT_LINE(ITEM.THE_ITEMS(1..ITEM.THE_LENGTH));

    END PUT_LINE;

    procedure PUT_LINE (FILE : in out TEXT_IO.FILE_TYPE;
                ITEM : in VAR_STRING) is
    - [procedure_header_comment]
    begin
```

```
     TEXT_IO.PUT_LINE (FILE => FILE, ITEM => STRING_VALUE(ITEM));
   end PUT_LINE;

   PROCEDURE GET_LINE(ITEM: OUT VAR_STRING) IS
--------------------------------------
- THE PURPOSE OF THIS PROCEDURE IS TO READ IN DATA AND TRANSFORM IT INTO
- THE VAR_STRING DATA TYPE
--------------------------------------

   WHOLE_LINE: STRING (1..80);  - USED TO READ IN THE ENTIRE LINE
   LINE_LENGTH: INTEGER;     - RECORDS THE AMOUNT OF CHARACTERS READ IN

   BEGIN
   - READ IN ENTIRE LINE OF CHARACTERS
   TEXT_IO.GET_LINE(WHOLE_LINE,LINE_LENGTH);

   - TRANSFER THE PERTINET DATA THROUGH STRING SLICES
   ITEM.THE_ITEMS(1..LINE_LENGTH) := WHOLE_LINE(1..LINE_LENGTH);

   - RECORD THE AMOUNT OF CHARACTERS READ IN
   ITEM.THE_LENGTH := LINE_LENGTH;

   END GET_LINE;

   procedure GET_LINE (FILE : in out TEXT_IO.FILE_TYPE;
                ITEM : out VAR_STRING) is
   - [procedure_header_comment]
   begin
     TEXT_IO.GET_LINE (FILE => FILE,
      ITEM => ITEM.THE_ITEMS,
      LAST => ITEM.THE_LENGTH);
   end GET_LINE;

   procedure OPEN (FILE : in out TEXT_IO.FILE_TYPE;
              MODE : in TEXT_IO.FILE_MODE;
              NAME : in VAR_STRING;
              FORM : in STRING := "") is
   - [procedure_header_comment]
   begin
     TEXT_IO.OPEN (
       FILE => FILE,
       MODE => MODE,
        NAME => NAME.THE_ITEMS(1..NAME.THE_LENGTH),
        FORM => FORM);
    end OPEN;

    procedure CREATE (FILE : in out TEXT_IO.FILE_TYPE;
            MODE : in TEXT_IO.FILE_MODE;
            NAME : in VAR_STRING;
            FORM : in STRING := "") is
    - [procedure_header_comment]
    begin
      TEXT_IO.CREATE (
        FILE => FILE,
        MODE => MODE,
        NAME => NAME.THE_ITEMS(1..NAME.THE_LENGTH),
        FORM => FORM);
    end CREATE;

    procedure ASSIGN (LEFT_VAR_STRING : out VAR_STRING; RIGHT_STRING : in STRING) is
    - [procedure_header_comment]
    begin
      if RIGHT_STRING'LAST = 0 then
        LEFT_VAR_STRING.THE_LENGTH := 0;
      else
    LEFT_VAR_STRING.THE_ITEMS(1..RIGHT_STRING'LAST) := RIGHT_STRING;
    LEFT_VAR_STRING.THE_LENGTH := RIGHT_STRING'LAST;
      end if;
    end ASSIGN;

    function IS_EQUAL (LEFT, RIGHT : in VAR_STRING) return BOOLEAN IS

    BEGIN

    - COMPARE JUST THE PERTINET PART OF THE STRING USING STRING SLICES
    IF LEFT.THE_ITEMS(1..LEFT.THE_LENGTH) = RIGHT.THE_ITEMS(1..RIGHT.THE_LENGTH)
THEN

    - RETURN TRUE WHEN EQUAL
    RETURN TRUE;
    ELSE

    - RETURN FALSE WHEN NOT EQUAL
    RETURN FALSE;
```

```
END IF;

END IS_EQUAL;

function IS_EQUAL (LEFT : in VAR_STRING; RIGHT : in STRING) return BOOLEAN IS

BEGIN

- COMPARE JUST THE PERTINET PART OF THE STRING USING STRING SLICES
IF LEFT.THE_ITEMS(1..LEFT.THE_LENGTH) = RIGHT THEN

- RETURN TRUE WHEN EQUAL
RETURN TRUE;
ELSE

- RETURN FALSE WHEN NOT EQUAL
RETURN FALSE;
END IF;

END IS_EQUAL;

 function "&" (LEFT, RIGHT : in VAR_STRING) return VAR_STRING is
 - [function_header_comment]
    DUMMY : VAR_STRING;
 begin
    if LEFT.THE_LENGTH = 0 then
       return RIGHT;
    elsif RIGHT.THE_LENGTH = 0 then
       return LEFT;
    else
 DUMMY := LEFT;
 DUMMY.THE_ITEMS(DUMMY.THE_LENGTH+1..DUMMY.THE_LENGTH+RIGHT.THE_LENG
TH) := RIGHT.THE_ITEMS(1..RIGHT.THE_LENGTH);
 DUMMY.THE_LENGTH := DUMMY.THE_LENGTH+RIGHT.THE_LENGTH;
 return DUMMY;
    end if;
 end "&";

 function "&" (LEFT : in STRING; RIGHT : in VAR_STRING) return VAR_STRING is
 - [function_header_comment]
    DUMMY : VAR_STRING;
 begin
```

```
  DUMMY.THE_ITEMS(1..LEFT'LAST) := LEFT;
  DUMMY.THE_LENGTH := LEFT'LAST;
  if RIGHT.THE_LENGTH /= 0 then
DUMMY := DUMMY&RIGHT;
  end if;
  return DUMMY;
end "&";

function "&" (LEFT : in VAR_STRING; RIGHT : in STRING) return VAR_STRING is
- [function_header_comment]
  DUMMY : VAR_STRING;
begin
  DUMMY.THE_ITEMS(1..RIGHT'LAST) := RIGHT;
  DUMMY.THE_LENGTH := RIGHT'LAST;
  if LEFT.THE_LENGTH /= 0 then
DUMMY := LEFT&DUMMY;
  end if;
  return DUMMY;
end "&";

function "&" (LEFT, RIGHT : in STRING) return VAR_STRING is
- [function_header_comment]
  DUMMY1,DUMMY : VAR_STRING;
begin
  DUMMY.THE_ITEMS(1..LEFT'LAST) := LEFT;
  DUMMY.THE_LENGTH := LEFT'LAST;
  DUMMY1.THE_ITEMS(1..RIGHT'LAST) := RIGHT;
  DUMMY1.THE_LENGTH := RIGHT'LAST;
  return DUMMY&DUMMY1;
end "&";

function "&" (LEFT : in VAR_STRING; RIGHT : in CHARACTER) return VAR_STRING is
- [function_header_comment]
  DUMMY : VAR_STRING;
begin
  DUMMY.THE_ITEMS(1) := RIGHT;
  DUMMY.THE_LENGTH := 1;
  if LEFT.THE_LENGTH /= 0 then
DUMMY := LEFT&DUMMY;
  end if;
  return DUMMY;
end "&";
```

```
    function "&" (LEFT : in STRING; RIGHT : in CHARACTER) return VAR_STRING is
    - [function_header_comment]
      DUMMY1,DUMMY : VAR_STRING;
    begin
      DUMMY.THE_ITEMS(1..LEFT'LAST) := LEFT;
      DUMMY.THE_LENGTH := LEFT'LAST;
      DUMMY1.THE_ITEMS(1) := RIGHT;
      DUMMY1.THE_LENGTH := 1;
      return DUMMY&DUMMY1;
    end "&";

    function STRING_VALUE (A_STRING : in VAR_STRING) return STRING is
    - [function_header_comment]
    begin
      return A_STRING.THE_ITEMS(1..A_STRING.THE_LENGTH);
    end STRING_VALUE;

    function STRING_VALUE (POSITION : INTEGER; A_STRING : in VAR_STRING) return CHAR-
ACTER is
    - [function_header_comment]
    begin
      return A_STRING.THE_ITEMS(POSITION);
    end STRING_VALUE;

    function STRING_LENGTH (A_STRING : in VAR_STRING) return INTEGER is
    - [function_header_comment]
    begin
      return A_STRING.THE_LENGTH;
    end STRING_LENGTH;
END STRING_OPERATIONS;
```

Bibliography

1. Kabrisky, Matthew. Class lecture, EENG 621, Pattern Recognition II. School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, April 1991.
2. Kabrisky, Matthew. Thesis Advisory meetings. School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, Sep 90 - Dec 91.
3. Parsons, Thomas W. *Voice and Speech Processing*. New York: McGraw Hill Book Company, 1987.
4. Barmore, Gary D. *Speech Recognition Using Neural Networks and Dynamic Programming*. Master's thesis, AFIT/GE/ENG-88D, School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, December 1988 (AD-A202528)
5. Recla, Wayne F. *A Study in Speech Recognition Using a Kohonen Neural Network, Dynamic Programming, and Multi-feature Fusion.* Master's thesis, AFIT/GE/ENG-89D, School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, December 1989 (AD-A216282).
6. Stowe, Francis Scott. *Speech Recognition Using a Kohonen Neural Networks, Dynamic Programming, and Multi-feature Fusion.* Master's thesis, AFIT/GE/ENG-90D-59, School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, December 1990.
7. Peacocke, Richard D. and Graf, Daryle H. "An Introduction to Speech and Speaker Recognition," *IEEE COMPUTER*, 26-33 (August 1990).
8. Rogers, Steven K. Class lecture, EENG 621, Pattern Recognition I. School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, February 1991.
9. Press, William H. and et al. *Numerical Recipes.* Cambridge University Press, 1986.
10. Lee, Kai-Fu, *Automatic Speech Recognition.* Klumer Academic Publisher, 1989.
11. Davis, Steven B. and Mermelstein, Paul. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Transaction of Accustic, Speech, and Signal Processing, 65-73 (1980).*
12. Radoy, Charles h. *Pattern Recognition by Fourier Series Transformation.* Master's thesis, AFIT/GE/EE-67A-11, School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, March 1967.
13. Rogers, Steven K. Thesis Advisory meetings. School of Engineering, Air Force Institude of Technology (AU), Wright-Patterson AFB OH, Sep 90 - Dec 91.